5101-274
Flat-Plate
Solar Array Project

DOE/JPL-1012-115
Distribution Category UC-63b

The SIMRAND I Computer Program

Simulation of Research and Development Projects

R.F. Miles, Jr.

February 15, 1986

Prepared for
U.S. Department of Energy
Through an Agreement with
National Aeronautics and Space Administration
by
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 85-96

5101-274
Flat-Plate
Solar Array Project

DOE/JPL-1012-115
Distribution Category UC-63b

The SIMRAND I Computer Program

Simulation of Research and Development Projects

R.F. Miles, Jr.

February 15, 1986

Prepared for
U.S. Department of Energy
Through an Agreement with
National Aeronautics and Space Administration
by
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 85-96

Prepared by the Jet Propulsion Laboratory, California Institute of Technology, for the U.S. Department of Energy through an agreement with the National Aeronautics and Space Administration.

The JPL Flat-Plate Solar Array Project is sponsored by the U.S. Department of Energy and is part of the National Photovoltaics Program to initiate a major effort toward the development of cost-competitive solar arrays.

This document reports on work done under NASA Task RE-152, Amendment 419, DOE/NASA IAA No. DE-AI01-85CE89008.

ABSTRACT

SIMulation of Research ANd Development Projects (SIMRAND) is a methodology for the selection of the optimal subset of systems or tasks to be implemented on a research and development (R&D) project. The SIMRAND methodology models the alternative subsets of systems or tasks under consideration with alternative networks. Each path through an alternative network represents one way of satisfying the project goals. Equations are developed that relate the system or task variables to the measure of preference. Uncertainty is incorporated by treating the variables of the equations probabilistically as random variables, with associated cumulative distribution functions. Cardinal utility functions are assessed over the measure of preference. The SIMRAND I Computer Program is run for each alternative network, incorporating the network structure, the equations, the cumulative distribution functions, and the utility functions in a Monte Carlo simulation model. The alternative network yielding the most-preferred value for the measure of preference, or the highest utility function value, is the optimal network.

This Report documents the SIMRAND I Computer Program (Version 5.0x03) written in Microsoft FORTRAN for the IBM PC microcomputer and its compatibles. The SIMRAND I Computer Program comprises eleven modules--a main routine and ten subroutines. Two additional files are used at compile time, one inserts the system or task equations into the source code, while the other inserts the dimension statements and common blocks. The SIMRAND I Computer Program can be run on most microcomputers or mainframe computers with only minor modifications to the computer code.

CONTENTS

SECTION 1

INTRODUCTION

SIMulation of Research ANd Development Projects (SIMRAND) is a methodology for the selection of the optimal subset of systems or tasks to be implemented on a research and development (R&D) project. This report documents the SIMRAND I Computer Program (Version 5.0x03).

The SIMRAND methodology uses alternative networks to model the alternative subsets of systems or tasks under consideration. Each path through an alternative network represents one way of satisfying the project goals. Equations are developed that relate the system or task variables to the measure of preference. Uncertainty is incorporated by treating the variables of the equations probabilistically as random variables, with associated cumulative distribution functions. Cardinal utility functions are assessed over the measure of preference. The SIMRAND I Computer Program is run for each alternative network, incorporating the network structure, the equations, the cumulative distribution functions, and the utility functions in a Monte Carlo simulation model. The alternative network yielding the most-preferred value for the measure of preference, or the highest utility function value, is the optimal network. An introduction to the SIMRAND methodology is given in Reference 1, while more extensive discussions of the methodology and implementation are presented in References 2 through 4.

The SIMRAND I Computer Program, in its present and earlier versions, has been used for the analysis of flat-plate solar-cell arrays (Reference 2 and 5), the analysis of solar-dish power systems (Reference 6), the analysis of spaceborne power systems, and the analysis of designs for spacecraft autonomy (Reference 7).

Following this introductory Section, Section 2 is a description of the SIMRAND I Computer Program. Section 3 discusses the SIMRAND I Computer Program code. Its present version, Version 5.0x03, is listed in Appendix A. Section 4 gives user instructions for the SIMRAND I Computer Program. Section 5 discusses the COSMIC (Computer Software Management & Information Center) microcomputer diskette, which contains the SIMRAND I Computer Program code. An example (Reference 2) of the input data file (DATAIN.DAT) for the SIMRAND I Computer Program is found in Appendix B. The output data file (DATAOUT.DAT), that would be generated from the input data file of Appendix B, is presented in Appendix C.

SECTION 2

PROGRAM DESCRIPTION

Version 5.0x03 of the SIMRAND I Computer Program is written in the Microsoft FORTRAN language. The Microsoft FORTRAN Compiler (References 8 and 9) is designed for the IBM PC microcomputer and its compatibles, which use the Intel 8086/8088/80186/80286 family of 16-bit microprocessors (Reference 10). The performance of the SIMRAND I Computer Program is greatly enhanced by the additional use of the Intel 8087/80287 family of numeric coprocessors. This version should run on most microcomputers or mainframe computers equipped with FORTRAN compilers, with only minor modification to the computer code.

The SIMRAND I Computer Program comprises eleven modules--a main routine and ten subroutines. Two additional files are "included" at compile time, one to insert the system or task equations into the source code (EQNS.FOR), and the other to insert all of the dimension statements and common blocks (INITIAL.FOR). The system or task equations, the dimension statements, and the common blocks used in Version 5.0x03 correspond to the SIMRAND model for the Large Scale Systems (LSS) publication (Reference 2). SIMRAND I Computer Program implementations for any application need only rewrite the two included files.

Figure 2-1 is the SIMRAND I Computer Program Tier Chart, similar in concept to that described by Tausworthe (Reference 11). The modules of the SIMRAND I Computer Program are numbered according to the "calling" module in the Program hierarchy. A brief description follows of the functions of each of the modules, with the modules numbered as shown in Figure 2-1:

1 MAIN

This module is the main routine for the SIMRAND I Computer Program. It initializes all the adjustable arrays. It is the main calling routine for the SIMRAND I Computer Program.

1.3 INPUT

Using the file DATAIN.DAT, this module inputs all the data for a SIMRAND I Computer Program run.

1.4 COEFF

This module calculates the Task Variable inverse cumulative distribution function (CDF) coefficients, the utility function coefficients, and the certainty equivalent coefficients from the input data.

1.5 RUNS

This module is the calling routine for the Monte Carlo runs.

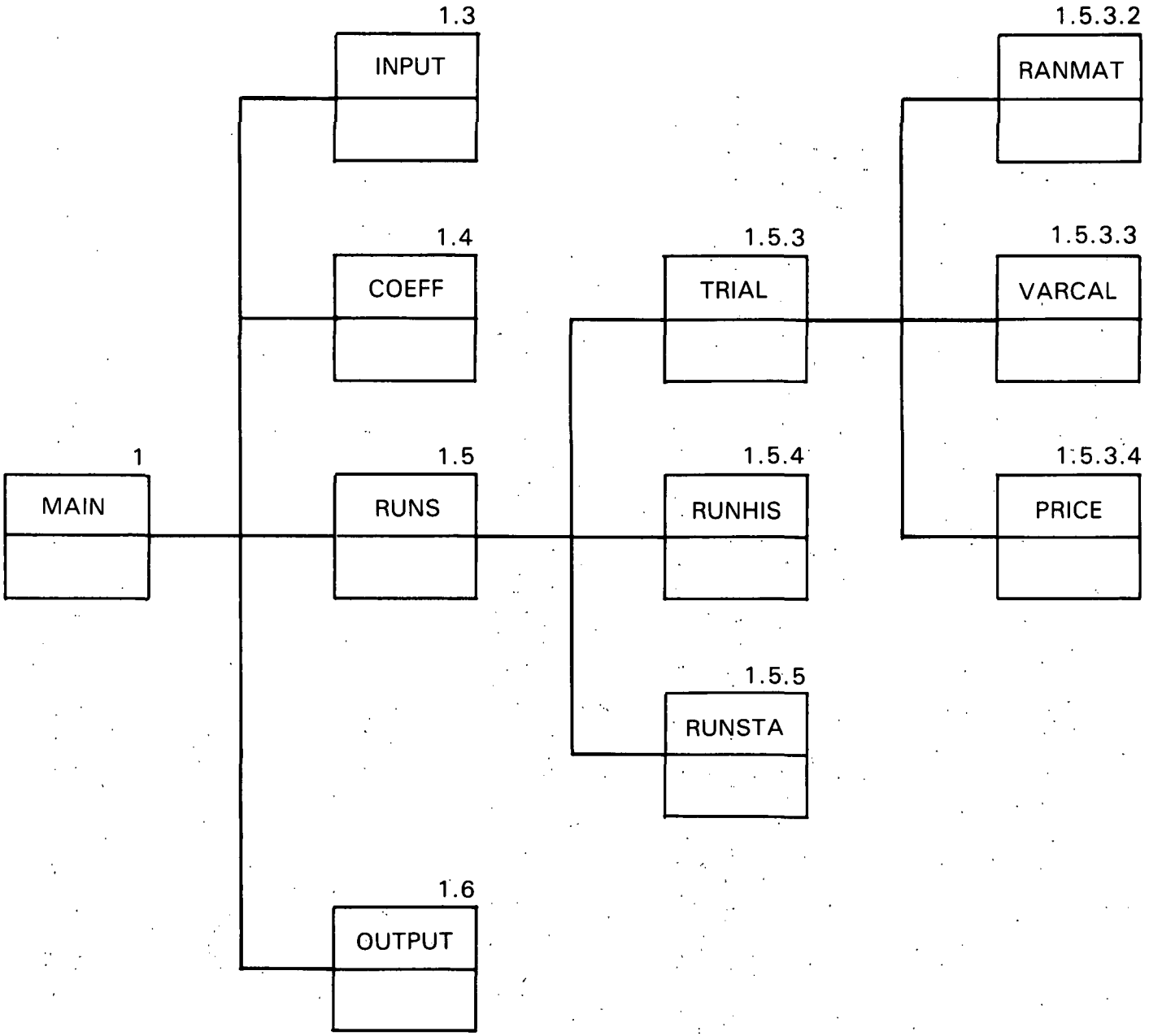

Figure 2-1. SIMRAND Ver. 5.0x03 Program Tier Chart

1.5.3 TRIAL

This module is the calling routine for a single Monte Carlo trial.

1.5.3.2 RANMAT

Calculation of the random number matrix for a single Monte Carlo trial is carried out by this module. The theory and the implementation of the random number generator are presented in Reference 4.

1.5.3.3 VARCAL

This module calculates the random variable matrix for a single Monte Carlo trial from the random number matrix using the Task Variable inverse CDF coefficients generated by the module COEFF. The random variables for this implementation of SIMRAND I are defined in Table 2-1.

1.5.3.4 PRICE

Using the random variable values generated by the module VARCAL, and the system or task equations, this module calculates the individual step performance measures for a single Monte Carlo trial. The system or task equations will be analogous to those defined for this implementation of SIMRAND I in Table 2-2. In the original applications of the SIMRAND I Computer Program (and in Reference 2), the step equations were prices, and thus this module was given the name PRICE.

1.5.4 RUNHIS

This module accumulates a history of the data generated by the module TRIAL.

1.5.5 RUNSTA

This module calculates the statistics for the SIMRAND I Program run.

1.6 OUTPUT

This module calculates percentile prices for the summary CDFs, calculates detailed CDFs for the individual step performance measures, and outputs all results to the file DATAOUT.DAT.

Table 2-1 depicts the LSS Price Variables for the Price Equations, while Table 2-2 shows the LSS Step Price and Total Price Equations, as used in Reference 2. As stated earlier, these equations will differ for other applications of the SIMRAND I Computer Program.

Table 2-3 lists the SIMRAND Array Indexing Variables, Table 2-4 lists the SIMRAND Local Variables, and Table 2-5 lists the SIMRAND Global Variables as defined in Common Blocks.

Table 2-1. SIMRAND LSS Price Variables for the Price Equations

I	Price Variable X(I)	Range	Symbol
1	Silicon Price	$/kg	SILICON
2	Cell Thickness	mils	T
3	Cell Efficiency	0 - 1.00	ηe
4	Crystallization Price	$/kg or $/$m^2$	CRYSTAL
5	Sawing Price	$/$m^2$	SAW
6	Cell Price	$/$m^2$	CELL
7	Include in Run	0/1	INCLUD

Table 2-2. SIMRAND LSS Step Price and Total Price Equations

STEP 1: Silicon Purification Production

P(1) = SILICON • T • CU • CD / CI • ηe

= (X(1)*X(2)*CU*CD) / (CI*X(3))

STEP 2: Crystallization Production

Ingot Technology (Paths 1 and 3)

P(2) = CRYSTAL • T • CU • CD / CI • ηe

= (X(4)*X(2)*CU*CD) / (CI*X(3))

Ribbon Technology (Paths 2 and 4)

P(2) = CRYSTAL / CI • ηe

= X(4) / (CI*X(3))

STEP 3: Sawing Production

Ingot Technology (Paths 1 and 3)

P(3) = SAW / CI • ηe

= X(5) / (CI*X(3))

Ribbon Technology (Paths 2 and 4)

P(3) = 0.0

STEP 4: Cell Production

P(4) = CELL / CI • ηe

= X(6) / (CI*X(3))

TOTAL PRICE:

P(5) = P(1) + P(2) + P(3) + P(4)

The three constants in the equations are:

Solar Insolation: CI = 1000 Wp/m^2

Silicon Density: CD = 2330 kg/m^3

Unit Conversion: CU = 2.54 x 10^{-5} m/mil

Table 2-3. SIMRAND Computer Program Version 5.0x03
Array Indexing Variables

Array Indexing Variable		Definition
IA	:	Indexing Variable for the Network PAths.
IAA	:	Indexing Variable for the Network PAths.
IC	:	Indexing Variable for the Coefficients of the piece-wise linear fits to the Task Variable Inverse CDF, Utility Functions, and Certainty Equivalents.
ID	:	Indexing Variable for the Task Variable CDF and Utility Input Data.
IH	:	Indexing Variable for the Step Price and Total Price Histograms, CDF, and PCDF.
IH25	:	Indexing Variable for writing Step Price and Total Price CDF and PCDF outputs. IH25 = IH+25.
II	:	Indexing Variable for the Individuals.
IP	:	Indexing Variable for the Step and Total Prices.
IT	:	Indexing Variable for the Task Variables.
IX	:	Indexing Variable for the Price Variables.
IZ	:	Indexing Variable for the Price Percentiles.

Table 2-4. SIMRAND Computer Program Version 5.0x03 Local Variables

Local Variable		Definition
DEL	:	Histogram Interval expressed in units of Price. DEL = (PUB(IP)-PLB(IP))/50.
F1	:	F1 = TDATA(IT,IC+3) or UDATA(II,IC+1).
F2	:	F2 = TDATA(IT,IC+5) or UDATA(II,IC+3).
IAMIN	:	Network Path identified during a Monte Carlo Trial as the Minimum Total Price Path.
IDTASK	:	IDTASK = ITASK(IA,IX).
IIH	:	IIH = IH.
INALT	:	0/1 Variable to indicate whether or not a Network Path has been initially selected during a Monte Carlo Trial.
INCLUD	:	Task Variable to indicate whether a given Network Path is to be included in the program run. INCLUD = ITASK(IA,NIX).
ITR	:	Identifier for the number of the Monte Carlo Trial.
NIAA	:	NIAA = IA - 1.
NICC	:	NICC = NIC - 1.
NIPP	:	NIPP = NIP - 1.
NIXX	:	NIXX = NIX - 1.
NIUU	:	NIUU = NIU - 2.
PERCNT	:	Running percentage in summing through Histogram.
RAN	:	RAN = RANMTX(IA,IX).
RANA	:	Multiplier Coefficient in the Random Number Generator.
RANC	:	Additive Coefficient in the Random Number Generator.
RANDIV	:	RANDIV = RANX/RANM.
RANM	:	Modulus for the Random Number Generator.
RANSUB	:	RANT*RANM.
RANT	:	DINT(RANDIV).
RANX	:	RANX = RANA*RANDOM + RANC.
RAN1	:	RAN1 = RAN/TDATA(IDTASK,2).
UCAL	:	Logical Variable set .TRUE. if Utility Value has been calculated.
V1	:	V1 = TDATA(IT,IC+2) or UDATA(II,IC).
V2	:	V2 = TDATA(IT,IC+4) or UDATA(II,IC).
X(IX)	:	X(IX) = VARMTX(IA,IX).

Table 2-5. SIMRAND Version 5.0x03 Global Variables as Defined in Common Blocks

Common Block "COM00":

IDATA	:	Integer Identifier for the Data for the SIMRAND Run.
JDIAG	:	Integer 0/1/2 variable. JDIAG = 0 for no diagnostic display during Run. JDIAG = 1 for minimum diagnostic display during Run. JDIAG = 2 for diagnostic display of all computations performed.
JLOUT	:	Integer 0/1 variable. JLOUT = 0 for summary statistics only. JLOUT = 1 for Long Output.
NTRIAL	:	Number of Monte Carlo Trials for the Run.
NIA	:	Number of PAths in the Task Network.
NIX	:	Number of Price Variables.
NIT	:	Number of Task Variables.
NID	:	Number of Data Points for the Task Variable CDFs.
NIP	:	Number of Step Prices and Total Price.
NII	:	Number of Individuals.
NIC	:	Number of Coefficients for the piece-wise linear fits to the Task Variable Inverse CDFs, the Utility Functions, and the Certainty Equivalents. NIC = NID - 4.
NIU	:	Number of Data Points for the Utility Function Input Data. NIU = NID - 2.

Common Block "COM01":

ITASK(IA,IX)	:	Integer Matrix identifying the Task Variable to be associated with each Price Variable (IX) for each Network Path (IA).
TDATA(IT,ID)	:	Input Data (ID) for the Task Variables (IT).
TCOEFF(IT,IC)	:	Coefficients (IC) for the piece-wise linear fits to the Task Variable Inverse CDF (IT).

Table 2-5. (Cont'd)

Common Block "COMO2":		
RNSEED	:	Number seed for the Random Number Generator.
RANDOM	:	Random Number generated by the Random Number Generator.
RANMTX(IA,IX)	:	Matrix of Random Numbers for Network Paths (IA) and Price Variables (IX).
VARMTX(IA,IX)	:	Matrix of Price Variables derived from RANMTX(IA,IX) and the Task Variable CDF. VARMTX(IA,IX) is "Equivalenced" to RANMTX(IA,IX).
Common Block "COMO3":		
UDATA(II,ID)	:	Input Data (ID) for the Utility Functions for Individuals (II).
UCOEF(II,IC)	:	Coefficients (IC) for the piece-wise linear fits to the Utility Functions for Individuals (II).
CECOEF(II,IC)	:	Coefficients (IC) for the piece-wise linear fits to the Certainty Equivalent Curves for Individuals (II).
Common Block "COMO4":		
PLB(IP)	:	Lower Bounds for the Step Price and Total Price (IP) Histograms.
PUB(IP)	:	Upper Bounds for the Step Price and Total Price (IP) Histograms.
Common Block "COMO5":		
PMEAN(IP)	:	Step Price and Total Price (IP) Means.
PMIN2(IP)	:	Intermediate calculations for Step Price and Total Price (IP) Variances.
PVAR(IP)	:	Variances for Step Prices and Total Price (IP).
PSDEV(IP)	:	Standard Deviations for Step Prices and Total Price (IP).

Table 2-5. (Cont'd)

Common Block "COM06":		
IHALT(IA)	:	Histogram of number of times a Network Path (IA) was selected during a SIMRAND Run.
IHPMIN(IP,IH)	:	Histograms (IH) for Step Prices and Total Price (IP).
Common Block "COM07":		
IITR	:	Identifier for a Monte Carlo Trial in a SIMRAND Run.
Common Block "COM08":		
UTIL(II)	:	Utility of the Network for Individuals (II) as calculated for a Monte Carlo Trial during a SIMRAND Run.
RUTIL(II)	:	Utility of the Network for Individuals (II) as calculated at the conclusion of a SIMRAND Run.
CERTEQ(II)	:	Certainty Equivalent of the Network for Individuals (II) as calculated for a SIMRAND Run.
Common Block "COM09":		
PMIN(IP)	:	The minimum Total Price and associated Step Prices (IP) calculated during a single Monte Carlo Trial of a SIMRAND Run.
PRMIN(IP)	:	The minimum Total Price and minimum Step Prices (IP) selected from PMIN(IP) for all Monte Carlo Trials during a SIMRAND Run.
PRMAX(IP)	:	The maximum Total Price and maximum Step Prices (IP) selected from PMIN(IP) for all Monte Carlo Trials during a SIMRAND Run.
PMTX(IA,IP)	:	Matrix of Step Prices and Total Prices (IP) for each Network Path (IA) calculated during a single Monte Carlo Trial of a SIMRAND Run.

Table 2-5. (Cont'd)

Common Block "COM10":

PRICEZ(IP,IZ)	:	Price for the (IZ) Percentile for the Step Prices and Total Price selected for PMIN(IP) for all Monte Carlo Trials during a SIMRAND Run.
CDF(IP,IH)	:	Cumulative Distribution Function over the Histogram Intervals (IH) for the Step Prices and Total Price (IP) selected for PMIN(IP) for all Monte Carlo Trials during a SIMRAND Run.
PCDF(IP,IH)	:	Step Price and Total Price (IP) for the Cumulative Distribution Function over the Histogram Intervals (IH) selected for PMIN(IP) for all Monte Carlo Trials during a SIMRAND Run.
ZPCT(IZ)	:	Probability to be used for the (IZ) Price Percentile.

SECTION 3

PROGRAM CODE

Appendix A gives the FORTRAN code for the present version, Version 5.0x03, of the SIMRAND I Computer Program. The EQNS.FOR file and the INITIAL.FOR file are written specifically for Reference 2. This version of the SIMRAND I Computer Program is written to run under the IBM DOS 2.1 Operating System (References 12 and 13) on an IBM PC compatible microcomputer with the Intel 8088 microprocessor and the Intel 8087 numeric coprocessor (Reference 10). The source code is to be compiled using the Microsoft FORTRAN Compiler (References 8 and 9). The SIMRAND I Computer Program can be compiled so as not to require the Intel 8087 numeric coprocessor, but the performance will suffer because the program is very numerically intensive. Other computer implementations will require minor modifications to the code. The compiled source-code files (object-code files) can be linked into an executable file, using either the Microsoft Linker provided with the Microsoft FORTRAN Compiler, or the Phoenix Software PLINK86 Linker (Reference 14), but not the IBM DOS 2.1 Linker.

The Microsoft FORTRAN Compiler calls two files external to the source code by use of the Microsoft FORTRAN INCLUDE statement. The Microsoft FORTRAN metacommand "$INCLUDE: 'INITIAL.FOR'" inserts as source code the INITIAL.FOR file, which includes all of the array dimension statements and all of the common block statements. Thus, when the SIMRAND I Computer Program is redimensioned for different size arrays, only one source code file needs to be changed. The Microsoft FORTRAN metacommand "$INCLUDE: 'EQNS.FOR'" inserts as source code the EQNS.FOR file, which includes all of the system or task equations. Thus, only two files need to be modified to rewrite the SIMRAND I Computer Program source code for any application.

SECTION 4

PROGRAM USER'S INSTRUCTIONS

An example (Reference 2) of the formatting of the input data, contained in a file always named DATAIN.DAT, is presented in Appendix B. The user should follow the column alignment of this example exactly, due to the nature of the FORTRAN READ Statement, especially with respect to integer variables. Most of the input data are entered into common blocks, and therefore appear as global variables to the SIMRAND I Program. The variable definitions are given in Tables 2-3, 2-4 and 2-5. More details concerning several of the variables are discussed as follows:

IDATA: This integer variable is an identifier for the run. Other than appearing in the output file it is not used by the SIMRAND I Computer Program.

JDIAG: This integer variable specifies the level of diagnostics to be displayed during a run. "JDIAG = 0" suppresses all diagnostic display during a run, and is particularly useful when the run is made in the background of a multi-tasking environment. "JDIAG = 1" displays all the input data as it is read from the DATAIN.DAT file, and also displays statistics after every 100 Monte Carlo trials. "JDIAG = 1" should be the preferred diagnostic level for runs made using single-tasking environments, or runs made in the foreground of multi-tasking environments. The diagnostics displayed will not degrade the performance of the run significantly, but will show sufficient information to indicate to the user that the run is proceeding correctly. "JDIAG = 2" should be used for debugging purposes only, as it displays the results of every intermediate calculation.

JLOUT: This integer variable specifies whether the DATAOUT.DAT output file is to be the short form (JDIAG = 0) or the long form (JDIAG = 1). In the short form, only statistics of the run are displayed. In the long form, in addition to the information of the short form, the CDFs of the step equations and their histograms are output. Appendix C is an example of the short form of the output.

NTRIAL: This integer variable specifies the number of Monte Carlo trials that are to be performed during the run. Typically useful numbers range from 100 to 10,000.

RNSEED: This double-precision real variable is the seed number for the random number generator, and may be any number between 0.0 and 33,554,431.0. When making repetitive runs with the same parameters to test the sensitivity of the results to the random number sequence used, enter the last random number generated (displayed in the output file) as the RNSEED for the next run. This will guarantee that the same, or a portion of the same random number sequence will not be used again. A total of 33,554,432 random numbers can be generated before the sequence repeats. The random number generator is a linear congruential generator with modulus 33,554,432. Other random number generators can easily be incorporated in the source code (Reference 4).

ITASK(IA,IX): This two-dimensional integer array defines the task network, and specifies the CDFs to be assigned to the random variables. The IA rows define alternative paths through the task network. The IX columns represent the random variables of the system or task equations. The elements of the array are the IT integers (IT = 1,...,NIT) that designate the CDFs to be used for the variables. The NIX variable (IX = 1,...,NIX) is always 0 or 1. If NIX = 0, it will exclude that IA alternative path from the network. If NIX = 1, it will include that IA alternative path in the network. This NIX feature is useful when making runs with subsets of the alternative paths, as the entire input file does not have to be rewritten.

TDATA(IT,ID): This two-dimensional real array gives the data for the CDFs, represented as the end-points of a piece-wise linear fit. Each IT row represents a different CDF. The numbers are paired up as: (random variable value, probability number). The first two numbers of each IT row (ID = 1,2) represent both the default random variable value if the task should fail, and the probability that the task will succeed. The following pairs of numbers (ID = 3,...,NID) represent points on the CDF, conditional on the task being successful. The random variable values must exhibit weak increasing monotonicity, i.e., they must never decrease in value, as is required for a mathematically proper CDF. The first number pair, starting with ID = 3, must have a probability number = 0.0. The last number pair used to define the CDF, which may be less than ID = NID, must have a probability number = 1.0. The program will accommodate any valid CDF, including not only ranges over which the probability value does not change, but also step functions representing discrete probability mass distributions.

UDATA(II,ID): This two-dimensional real array gives the data for the utility functions of NII (II = 1,...,NII) decision makers. It represents the end-points of a piece-wise linear fit (ID = 1,...,NID). At least one utility function must be entered for the SIMRAND I Computer Program to input data properly, even if the utility data is not relevant to the specific application. The utility function can have any shape, but it must be monotonic (either increasing or decreasing) for its inverse (the certainty equivalent) to be meaningful. This is not a shortcoming of the SIMRAND I Computer Program, but is a property inherent in the definition of "certainty equivalent."

PLB(IP) AND PUB(IP): These one-dimensional real arrays define the lower and upper bounds for the CDFs for the step equations as given in the long-form output (JLOUT = 1) of the DATAOUT.DAT output file. There are no restrictions on these bounds, and they may be narrowed to examine parts of the CDFs in detail. The SIMRAND I Computer Program divides the alternative networks into a series of steps or stages (IP = 1,...,NIP), as that is a form that is typically useful for most applications. The NIP Step is reserved for the performance measure of the alternatives, and is used in selecting the most preferred alternative for each Monte Carlo trial. In the original applications of the SIMRAND I Computer Program (and in Reference 2), the step equations were prices, with the NIP Step being the sum of the previous steps, and thus the indices were defined with the letter "P".

The output of the SIMRAND I Computer Program is always written to the DATAOUT.DAT file. An example of the short form of the output file is given in Appendix C. It gives both some input information for identification purposes

(IDATA, NTRIAL, and RANDOM SEED) and the means and standard deviations of the CDF values for the step equations. It yields seven points on the CDFs for each of the step equations (minimum, maximum, and the fractiles: 0.10, 0.25, 0.50, 0.75, and 0.90). For the run for each decision maker (II = 1,...,NII), it presents the utility function value as well as the certainty equivalent (the inverse function value of the utility function value). It yields the histogram data (IHALT(IA)) for all the alternative paths as defined for the run in the ITASK(IA,IX) array. The IHALT(IA) data state the number of times that the Monte Carlo run selected each of the IA alternative paths as the optimal path. It gives the random number seed (RNSEED) to be used for the next run in a series of repetitive runs with different random number seeds. The long form of the output file (JLOUT = 1) gives 51 points on the CDFs for each of the step equations.

SECTION 5

COSMIC DISKETTE

The SIMRAND I Computer Program is available on microcomputer diskette from COSMIC (Computer Software Management & Information Center), NASA's clearinghouse where software is transferred from government agencies to industrial or other users (Reference 15). The microcomputer diskette is an industry-standard 5-1/4 inch, double-sided, double-density, soft-sector diskette, with 40 tracks and 9 sectors per track. It can be read with the Microsoft MS-DOS (Version 2.0 or later) operating system (References 8 and 9).

The COSMIC diskette contains all the source code files for the SIMRAND I Computer Program: COEFF.FOR, EQNS.FOR, INITIAL.FOR, INPUT.FOR, MAIN.FOR, OUTPUT.FOR, PRICE.FOR, RANMAT.FOR, RUNHIS.FOR, RUNS.FOR, RUNSTA.FOR, TRIAL.FOR, and VARCAL.FOR. It contains the input and output files DATAIN.DAT and DATAOUT.DAT with the data for the Large Scale Systems paper (Reference 2), with the one exception that that the number of Monte Carlo trials (NTRIAL) is set to 1,000 rather than 10,000. The diskette contains a microcomputer executable file, SIMRAND.EXE, which requires the use of the Intel 8087 numeric coprocessor. The diskette contains two batch processing files for compiling (CMPLSIM.BAT) and linking (LINKSIM.BAT) the source code files into executable form, and two files (LINKOBJ.LNK and LINKLIB.LNK) for supporting the linking process. All of these files, with the exception of the DATAOUT.DAT file, are read-only protected. The SIMRAND I Computer Program can be executed by typing "SIMRAND" at the operating system prompt.

The COSMIC diskette also contains several auxiliary files for the random number generator used by the SIMRAND I Computer Program. These auxiliary files permit the user to examine the performance of the random number generator if its parameters are changed. The theory and application of the random number generator, and the auxiliary files, are discussed in Reference 4.

SECTION 6

REFERENCES

1. Miles, R. F., Jr., Introduction to SIMRAND: SIMulation of Research ANd Development Projects, JPL Publication 82-20, Jet Propulsion Laboratory, Pasadena, California, March 1, 1982.

2. Miles, R. F., Jr., "The SIMRAND Methodology: Simulation of Research and Development Projects," Large Scale Systems, Vol. 7, pp. 59-67, 1984.

3. Miles, R. F., Jr., The SIMRAND Methodology: Theory and Application for the Simulation of Research and Development Projects, JPL Publication 85-98, Jet Propulsion Laboratory, Pasadena, California, February 15, 1986.

4. Miles, R. F., Jr., The RANDOM Computer Program: A Linear Congruential Random Number Generator, JPL Publication 85-97, Jet Propulsion Laboratory, Pasadena, California, February 15, 1986.

5. Borden, C. S., and Schwartz, D. L., Relative Potentials of Concentrating and Two-Axis Tracking Flat-Plate Photovoltaic Arrays for Central-Station Applications, JPL Publication 85-16, Jet Propulsion Laboratory, Pasadena, California, December 31, 1984.

6. Balbien, J., Probabilistic Cost Study of Solar Disk Power Systems, JPL Document No. 900-990, Jet Propulsion Laboratory, Pasadena, California, March 1981.

7. Smith, J. H., Autonomy Evaluation Methodology: A Microcomputer Tool for Design Tradeoffs, Vol. I, II, and III, JPL Document D-2761, Jet Propulsion Laboratory, Pasadena, California, November 1, 1985.

8. Microsoft FORTRAN Compiler for the MS-DOS Operating System: User's Guide, Version 3.30, Microsoft Corporation, Bellevue, Washington, 1985.

9. Microsoft FORTRAN: Reference Manual, Microsoft Corporation, Bellevue, Washington, 1985.

10. iAPX 86/88, 186/188 User's Manual: Programmer's Reference, Intel Corporation, Santa Clara, California, 1985.

11. Tausworthe, R. C., Standardized Development of Computer Software, Part I: Methods and Part II: Standards, Prentice-Hall, Englewood Cliffs, New Jersey, Part I in 1977 and Part II in 1979.

12. Microsoft Corp., Disk Operating System, Version 2.10 (First Edition), IBM No. 1502343, International Business Machines, Boca Raton, Florida, September 1983.

13. Microsoft Corp., Disk Operating System: Technical Reference, Version 2.10 (First Edition), IBM No. 1502346, International Business Machines, Boca Raton, Florida, September 1983.

PAGE 5-2 INTENTIONALLY BLANK

14. PLINK86: Linkage Editor for Intel 8086/8088, Phoenix Software Associates Ltd., North Easton, Maine, Ver. 1.47, 1985.

15. COSMIC, Suite 112, Barrow Hall, The University of Georgia, Athens, Georgia 30602, Phone: (404) 542-3265.

APPENDIX A

COMPUTER PROGRAM CODE

File: MAIN.FOR

APPENDIX A
Computer Program Code

```
************************************************************************
*                          PROGRAM SIMRAND I                           *
*                                                                      *
*  THIS IS THE PROGRAM SIMRAND I: SIMULATION OF RESEARCH AND           *
*  DEVELOPMENT PROJECTS.  IT IS WRITTEN IN MICROSOFT MS-DOS FORTRAN    *
*  VER. 3.2.  IN GENERAL, FORTRAN PROCESSORS WILL REQUIRE THAT MINOR   *
*  MODIFICATIONS BE MADE TO THE PROGRAM.                               *
*                                                                      *
*  PROGRAMMER: RALPH F. MILES, JR.                                     *
*              SYSTEMS DIVISION                                        *
*              JET PROPULSION LABORATORY                               *
*              PASADENA, CALIFORNIA 91109                              *
*                                                                      *
*  VERSION 5.0x03.                                                     *
*  DATE:   04/09/85                                                    *
*                                                                      *
*----------------------------------------------------------------------*
*                          CONFIGURATION                               *
*                                                                      *
*  VERSION    DATE                  CHANGES                           *
*                                                                      *
*  5.0X01   01/18/85   * IBM ANSI FORTRAN-77 VERSION DERIVED FROM      *
*                        IBM ANSI FORTRAN-66 VER. 4.0X1.               *
*  5.0X02   01/22/85   * RUNSTA.FOR (MODULE 1.5.4)                     *
*                      * OUTPUT.FOR (MODULE 1.6)                       *
*  5.0X03   04/09/85   * MAIN.FOR   (MODULE 1)                         *
*                      * RUNS.FOR   (MODULE 1.5)                       *
*                      * TRIAL.FOR  (MODULE 1.5.3)                     *
*                      * RANMAT.FOR (MODULE 1.5.3.2)                   *
*                      * VARCAL.FOR (MODULE 1.5.3.3)                   *
*                      * PRICE.FOR  (MODULE 1.5.3.4)                   *
*                      * RUNHIS.FOR (MODULE 1.5.4)                     *
*                      * RUNSTA.FOR (MODULE 1.5.5)                     *
*                                                                      *
************************************************************************


************************************************************************
*                            MAIN.FOR                                  *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1                                                          *
*  DATE:    01/31/85                                                   *
*                                                                      *
*  THIS IS THE MAIN ROUTINE OF THE PROGRAM SIMRAND I.  IT IS THE       *
*  ENTRY POINT FOR THE PROGRAM AND THE CALLING ROUTINE FOR THE FIRST   *
*  LEVEL OF SUBROUTINES.                                               *
*                                                                      *
*----------------------------------------------------------------------*
*                      CONFIGURATION CHANGES                           *
*                                                                      *
*    DATE                    CHANGE                                    *
*                                                                      *
*  12/05/84  * ORIGINAL.                                               *
```

File: MAIN.FOR

```
*  01/31/85  * MODULE  1: ALWAYS DISPLAY 'START MAIN ROUTINE'.          *
*            * MODULE  7: ALWAYS DISPLAY 'STOP  MAIN ROUTINE'.          *
*                                                                       *
*************************************************************************

$TITLE:'MAIN.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

*************************************************************************

      PROGRAM SIMRAN

*************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      OPEN (1,FILE='DATAIN.DAT' ,STATUS='OLD')
      OPEN (2,FILE='DATAOUT.DAT',STATUS='NEW')

***   {END MODULE 1}


***** READ RUN PARAMETERS.  {MODULE 2}

      READ   (1,100) IDATA,JDIAG,JLOUT,NTRIAL,NIA,NIX,NIT,NID,NIP,NII
100   FORMAT (/10I5)

         WRITE  (*,110)
110      FORMAT (/1X,'START MAIN ROUTINE')

*     READ RANDOM NUMBER SEED.
      READ   (1,120) RNSEED
120   FORMAT (/F15.0)

      RANDOM = RNSEED

      NIC = NID - 4
      NIU = NID - 2

      IF (JDIAG .NE. 0) THEN

         WRITE  (*,130) IDATA,JDIAG,JLOUT,NTRIAL,
     *                  NIA,NIX,NIT,NID,NIP,NII,NIC,NIU
130      FORMAT (/1X,'IDATA JDIAG JLOUT NTRIAL NIA NIX NIT NID  NIP',
     *           '  NII  NIC  NIU'/1X,12I5)

         WRITE  (*,140) RNSEED
140      FORMAT (1X,'RANDOM NUMBER SEED = ',F15.0)
```

```
      ENDIF

***   {END MODULE 2}


***** CALL SUBROUTINES.  {MODULES 3 - 6}

*     CALL MODULE 1.3.  {MODULE 3}
      CALL INPUT

*     CALL MODULE 1.4.  {MODULE 4}
      CALL COEFF

*     CALL MODULE 1.5.  {MODULE 5}
      CALL RUNS

*     CALL MODULE 1.6.  {MODULE 6}
      CALL OUTPUT

***  {END MODULES 3 - 6}


***** STOP PROGRAM SIMRAND.  {MODULE 7}

         WRITE (*,999)
999      FORMAT (/1X,'STOP MAIN ROUTINE')

      STOP

      END

***  {END MODULE 7}

****************************** MAIN.FOR ***********************************
```

```
************************************************************************
*                            INPUT.FOR                                 *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.3                                                        *
*  DATE:    12/05/84                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE INPUT.  IT READS THE INPUT DATA FROM THE     *
*  INPUT DATA FILE "DATAIN.DAT" (UNIT #1).                             *
*                                                                      *
*----------------------------------------------------------------------*
*                         CONFIGURATION CHANGES                        *
*                                                                      *
*    DATE                        CHANGE                                *
*                                                                      *
*  12/05/84  * ORIGINAL.                                               *
*                                                                      *
************************************************************************

$TITLE:'INPUT.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE INPUT

************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .NE. 0) THEN
         WRITE (*,100)
100      FORMAT (/1X,'ENTER SUBROUTINE INPUT')
      ENDIF

***   {END MODULE 1}


***** READ & WRITE TASK IDENTIFICATION MATRIX ITASK(IA,IX).  {MODULE 2}

      IF (JDIAG .NE. 0) THEN
         WRITE  (*,*) 'TASK IDENTIFICATION MATRIX ITASK(IA,IX)'
      ENDIF

      READ   (1,110)
110   FORMAT (1X)

      DO 150 IA=1,NIA
```

```
         READ   (1,120) (ITASK(IA,IX),IX=1,NIX)
120      FORMAT (10X,10I5)

         IF (JDIAG .NE. 0) THEN

            IF (NIX .LE. 10) THEN
               WRITE  (*,130) IA,(ITASK(IA,IX),IX=1,NIX)
            ELSE
               WRITE  (*,130) IA,(ITASK(IA,IX),IX=1,10)
               WRITE  (*,140)    (ITASK(IA,IX),IX=11,NIX)
            ENDIF

130         FORMAT (1X,I5,5X,10I5)
140         FORMAT (1X,10X,  10I5)

         ENDIF

150   CONTINUE

***   {END MODULE 2}


***** READ & WRITE TASK VARIABLE DATA MATRIX TDATA(IT,ID).  {MODULE 3}

      IF (JDIAG .NE. 0) THEN
         WRITE  (*,*) 'TASK VARIABLE DATA MATRIX TDATA(IT,ID).'
      ENDIF

      READ   (1,110)

      DO 190 IT=1,NIT

         READ   (1,110)

         READ   (1,160) (TDATA(IT,ID),ID=1,NID)
160      FORMAT (8X,4E14.4)

         IF (JDIAG .NE. 0) THEN

            WRITE  (*,170) IT,(TDATA(IT,ID),ID=1,4)
            WRITE  (*,180)    (TDATA(IT,ID),ID=5,NID)

170         FORMAT (1X,I5,3X,1P4E14.4)
180         FORMAT (1X,8X,   1P4E14.4)

         ENDIF

190   CONTINUE

***   {END MODULE 3}


***** READ & WRITE UTILITY DATA MATRIX UDATA(II,ID).  {MODULE 4}
```

```
      IF (JDIAG .NE. 0) THEN
         WRITE  (*,*) 'UTILITY DATA MATRIX UDATA(II,ID)'
      ENDIF

      READ   (1,110)

      DO 200 II=1,NII

         READ   (1,160) (UDATA(II,ID),ID=1,NIU)

         IF (JDIAG .NE. 0) THEN

            IF (NIU .EQ. 4) THEN
               WRITE  (*,170) II,(UDATA(II,ID),ID=1,4)
            ELSE
               WRITE  (*,170) II,(UDATA(II,ID),ID=1,4)
               WRITE  (*,180)    (UDATA(II,ID),ID=5,NIU)
            ENDIF

         ENDIF

200   CONTINUE

***   {END MODULE 4}


***** READ & WRITE PRICE RANGE BOUNDS PLB(IP) & PUB(IP).  {MODULE 5}

      IF (JDIAG .NE. 0) THEN
         WRITE  (*,*) 'PRICE RANGE BOUNDS PLB(IP) & PUB(IP).'
      ENDIF

*     READ & WRITE LOWER BOUND PRICES PLB(IP).

      READ   (1,110)

      READ   (1,210) (PLB(IP),IP=1,NIP)
210   FORMAT (8X,5E14.4)

      IF (JDIAG .NE. 0) THEN

         IF (NIP .LE. 4) THEN
            WRITE  (*,220) (PLB(IP),IP=1,NIP)
         ELSE
            WRITE  (*,220) (PLB(IP),IP=1,4)
            WRITE  (*,230) (PLB(IP),IP=5,NIP)
         ENDIF

220      FORMAT (1X,'PLB',5X,1P5E14.4)
230      FORMAT (1X,8X,      1P5E14.4)

      ENDIF

*     READ & WRITE UPPER BOUND PRICES PLB(IP).
```

File: INPUT.FOR

```
      READ   (1,110)

      READ   (1,210) (PUB(IP),IP=1,NIP)

      IF (JDIAG .NE. 0) THEN

         IF (NIP .LE. 4) THEN
            WRITE  (*,240) (PUB(IP),IP=1,NIP)
         ELSE
            WRITE  (*,240) (PUB(IP),IP=1,4)
            WRITE  (*,250) (PUB(IP),IP=5,NIP)
         ENDIF

240      FORMAT (1X,'PUB',5X,1P5E14.4)
250      FORMAT (1X,      8X,1P5E14.4)

      ENDIF

***   {END MODULE 5}


***** EXIT SUBROUTINE INPUT.  {MODULE 6}

      IF (JDIAG .NE. 0) THEN
         WRITE (*,999)
999      FORMAT (1X,'EXIT SUBROUTINE INPUT')
      ENDIF

      RETURN

      END

***   {END MODULE 6}

****************************** INPUT.FOR ***********************************
```

File: COEFF.FOR

```
************************************************************************
*                              COEFF.FOR                               *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.4                                                        *
*  DATE:    12/05/84                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE COEFF.  IT CALCULATES THE CDF COEFFICENTS,   *
*  THE UTILITY FUNCTION COEFFICIENTS, AND THE CERTAINTY EQUIVALENT     *
*  COEFFICIENTS FROM THE INPUT DATA.                                   *
*                                                                      *
*----------------------------------------------------------------------*
*                          CONFIGURATION CHANGES                       *
*                                                                      *
*    DATE                          CHANGE                            *
*                                                                      *
*  12/05/85  * ORIGINAL.                                               *
*                                                                      *
************************************************************************

$TITLE:'COEFF.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE COEFF

************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .NE. 0) THEN
         WRITE (*,100)
100      FORMAT (/1X,'ENTER SUBROUTINE COEFF')
      ENDIF

***   {END MODULE 1}


****C CALCULATE TASK VARIABLE COEFFICIENTS TCOEF(IT,IC).  {MODULE 2}

*     CALCULATE TCOEF(IT,IC) & TCOEF(IT,IC+1)
*     FOR VARMTX(IA,IX) = TCOEF(IT,IC)*RANMTX(IA,IX) + TCOEF(IT,IC+1)

      NICC = NIC - 1
      DO 130 IT=1,NIT
         DO 130 IC=1,NICC,2

            V1 = TDATA(IT,IC+2)
```

```
            V2 = TDATA(IT,IC+4)
            F1 = TDATA(IT,IC+3)
            F2 = TDATA(IT,IC+5)

            IF (F2 .EQ. F1) THEN
               TCOEF(IT,IC)   = 0.0
               TCOEF(IT,IC+1) = 0.0
            ELSE
               TCOEF(IT,IC)   = (V2-V1)/(F2-F1)
               TCOEF(IT,IC+1) = V1 - F1*TCOEF(IT,IC)
            ENDIF

130   CONTINUE

***   {END MODULE 2}


***** START (II,IC) DO LOOP TO CALCULATE UTILITY FUNCTION AND
***** CERTAINTY EQUIVALENT COEFFICIENTS.  {MODULE 3}

      NIUU = NIU - 2
      DO 170 II=1,NII
         DO 170 IC=1,NIUU,2


*****       CALCULATE UTILITY FUNCTION COEFFICIENTS.  {MODULE 4}

*           CALCULATE UCOEF(II,IC) & UCOEF(II,IC+1)
*           FOR UTIL(II) = UCOEF(II,IC)*PMIN(NIP) + UCOEF(II,IC+1)

            V1 = UDATA(II,IC)
            V2 = UDATA(II,IC+2)
            F1 = UDATA(II,IC+1)
            F2 = UDATA(II,IC+3)

            IF (V1 .EQ. V2) THEN
               UCOEF(II,IC)   = 0.0
               UCOEF(II,IC+1) = 0.0
            ELSE
               UCOEF(II,IC)   = (F2 - F1)/(V2 - V1)
               UCOEF(II,IC+1) = F1 - V1*UCOEF(II,IC)
            ENDIF

***         {END MODULE 4}


*****       CALCULATE CERTAINTY EQUIVALENT COEFFICIENTS.  {MODULE 5}

*           CALCULATE CECOEF(II,IC) & CECOEF(II,IC+1)
*           FOR CERTEQ(II) = CECOEF(II,IC)*RUTIL(II) + CECOEF(II,IC+1)

            IF (UCOEF(II,IC) .EQ. 0.0) THEN
               CECOEF(II,IC)   = 0.0
               CECOEF(II,IC+1) = 0.0
```

```
            ELSE
               CECOEF(II,IC)   =               1/UCOEF(II,IC)
               CECOEF(II,IC+1) = -UCOEF(II,IC+1)/UCOEF(II,IC)
            ENDIF

***         {END MODULE 5}

170   CONTINUE

***   {END MODULE 3}


***** WRITE TASK VARIABLE COEFFICIENTS TCOEF(IT,IC).  {MODULE 6}

      IF (JDIAG .EQ. 2) THEN

         WRITE (*,*)  'TASK VARIABLE COEFFICIENTS TCOEF(IT,IC)'

         DO 200 IT=1,NIT

            IF (NIC .EQ. 4) THEN
               WRITE  (*,180) IT,(TCOEF(IT,IC),IC=1,NIC)
            ELSE
               WRITE  (*,180) IT,(TCOEF(IT,IC),IC=1,4)
               WRITE  (*,190)    (TCOEF(IT,IC),IC=5,NIC)
            ENDIF

180         FORMAT (1X,I5,3X,1P4E14.4)
190         FORMAT (1X,   8X,1P4E14.4)

200      CONTINUE

      ENDIF

***   {END MODULE 6}


***** WRITE UTILITY FUNCTION COEFFICENTS UCOEF(II,IC).  {MODULE 7}

      IF (JDIAG .EQ. 2) THEN

         WRITE  (*,*) 'UTILITY FUNCTION COEFFICIENTS UCOEF(II,IC)'

         DO 210 II=1,NII

            IF (NIC .EQ. 4) THEN
               WRITE  (*,180) II,(UCOEF(II,IC),IC=1,NIU)
            ELSE
               WRITE  (*,180) II,(UCOEF(II,IC),IC=1,4)
               WRITE  (*,190)    (UCOEF(II,IC),IC=5,NIU)
            ENDIF

210      CONTINUE
```

```
      ENDIF

***   {END MODULE 7}

***** WRITE CERTAINTY EQUIVALENT COEFFICIENTS CECOEF(II,IC).  {MODULE 8}

      IF (JDIAG .EQ. 2) THEN

         WRITE  (*,*) 'CERTAINTY EQUIVALENT COEFFICIENTS CECOEF(II,IC)'

         DO 220 II=1,NII

            IF (NIC .EQ. 4) THEN
               WRITE  (*,180) II,(CECOEF(II,IC),IC=1,NIU)
            ELSE
               WRITE  (*,180) II,(CECOEF(II,IC),IC=1,4)
               WRITE  (*,190)    (CECOEF(II,IC),IC=5,NIU)
            ENDIF

220      CONTINUE

      ENDIF

***   {END MODULE 8}

***** EXIT SUBROUTINE COEFF.  {MODULE 9}

      IF (JDIAG .NE. 0) THEN
         WRITE (*,999)
999      FORMAT (1X,'EXIT  SUBROUTINE COEFF')
      ENDIF

      RETURN

      END

***  {END MODULE 9}

************************** COEFF.FOR ************************************C
```

File: RUNS.FOR

```
************************************************************************
*                              RUNS.FOR                                *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.5                                                        *
*  DATE:    02/05/84                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE RUNS.  IT IS THE CALLING SUBROUTINE FOR THE  *
*  MONTE CARLO CALCULATIONS.                                           *
*                                                                      *
*----------------------------------------------------------------------*
*                         CONFIGURATION CHANGES                        *
*                                                                      *
*    DATE                        CHANGE                                *
*                                                                      *
*  12/04/84  * ORIGINAL.                                               *
*  02/05/85  * MODULE  1: DO 130 IH= 1,52 (NOT 53).                    *
*                                                                      *
************************************************************************


$TITLE:'RUNS.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE RUNS

************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .NE. 0) THEN
         WRITE  (*,100)
100      FORMAT (/1X,'ENTER SUBROUTINE RUNS')
      ENDIF

      DO 110 IP=1,NIP
         PMIN2(IP) = 0.0
         PVAR(IP)  = 0.0
110   CONTINUE

      DO 120 IA=1,NIA
         IHALT(IA) = 0
120   CONTINUE

      DO 130 IP=1,NIP
         DO 130 IH=1,52
            IHPMIN(IP,IH) = 0
```

```
130   CONTINUE

      DO 140 II=1,NII
         RUTIL(II) = 0.0
140   CONTINUE

***   {END MODULE 1}


***** DO LOOP FOR TRIALS.  {MODULE 2}

      DO 160 ITR=1,NTRIAL

         IITR = ITR

         IF ((JDIAG .EQ. 1 .AND. MOD(ITR,100) .EQ. 0) .OR.
     *       (JDIAG .EQ. 2)) THEN
            WRITE  (*,150) ITR
150         FORMAT (/1X,'ITR = ',I5)
         ENDIF


*****    CALCULATE PRICES FOR ONE TRIAL.  {MODULE 3}

         CALL TRIAL

***      {END MODULE 3}


*****    INCREMENT HISTOGRAMS.  {MODULE 4}

         CALL RUNHIS

***      {END MODULE 4}


*****    ACCUMULATE STATISTICS.  {MODULE 5}

         CALL RUNSTA

***      {END MODULE 5}

160   CONTINUE

***   {END MODULE 2}


***** EXIT FROM SUBROUTINE RUNS.  {MODULE 6}

      IF (JDIAG .NE. 0) THEN
         WRITE  (*,999)
999      FORMAT (1X,'EXIT SUBROUTINE RUNS')
      ENDIF
```

```
      RETURN
      END
***   {END MODULE 6}
****************************** RUNS.FOR ************************************
```

File: TRIAL.FOR

```
************************************************************************
*                            TRIAL.FOR                                 *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.5.3                                                      *
*  DATE:    01/29/85                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE TRIAL.  IT IS THE CALLING SUBROUTINE FOR A   *
*  SINGLE MONTE CARLO TRIAL.                                           *
*                                                                      *
*----------------------------------------------------------------------*
*                     CONFIGURATION CHANGES                            *
*                                                                      *
*    DATE                    CHANGE                                    *
*                                                                      *
*  12/04/84   * ORIGINAL.                                              *
*  01/29/85   * MODULE  1: JDIAG .EQ. 2.                               *
*             * MODULE  5: JDIAG .EQ. 2.                               *
*                                                                      *
************************************************************************

$TITLE:'TRIAL.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE TRIAL

************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .EQ. 2) THEN
         WRITE  (*,100)
100      FORMAT (/1X,2X,'ENTER SUBROUTINE TRIAL')
      ENDIF

***   {END MODULE 1}


***** CALL SUBROUTINES FOR SINGLE MONTE CARLO TRIAL.  {MODULES 2-4}

*     CALL MODULE 1.5.3.2.  {MODULE 2}
      CALL RANMAT

*     CALL MODULE 1.5.3.3.  {MODULE 3}
      CALL VARCAL

*     CALL MODULE 1.5.3.4.  {MODULE 4}
```

File: TRIAL.FOR

```
      CALL PRICE

***   {END MODULES 2-4}


***** EXIT FROM SUBROUTINE TRIAL.  {MODULE 5}

      IF (JDIAG .EQ. 2) THEN
         WRITE  (*,999)
999      FORMAT (1X,2X,'EXIT SUBROUTINE TRIAL')
      ENDIF

      RETURN

      END

***  {END MODULE 5}

***************************** TRIAL.FOR ***********************************
```

File: RANMAT.FOR

```
*************************************************************************
*                          RANMAT.FOR                                   *
*                                                                       *
*  PROGRAM: SIMRAND I                                                   *
*  MODULE:  1.5.3.2                                                     *
*  DATE:    01/29/85                                                    *
*                                                                       *
*  THIS IS THE SUBROUTINE RANMAT.  IT CALCULATES THE RANDOM NUMBER      *
*  MATRIX FOR A SINGLE MONTE CARLO TRIAL.                               *
*                                                                       *
*-----------------------------------------------------------------------*
*                  CONFIGURATION CHANGES                                *
*                                                                       *
*    DATE                CHANGE                                         *
*                                                                       *
*  01/18/85  * ORIGINAL.                                                *
*  01/29/85  * MODULE  1: JDIAG .EQ. 2.                                 *
*            * MODULE  8: JDIAG .EQ. 2.                                 *
*                                                                       *
*************************************************************************

$TITLE:'RANMAT.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

*************************************************************************

      SUBROUTINE RANMAT

*************************************************************************


***** INITIALIZE.  {MODULE 1}

      DOUBLE PRECISION RANA,RANC,RANM,RANX,RANDIV,RANT,RANSUB
      LOGICAL          RANCAL

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .EQ. 2) THEN
         WRITE (*,100)
100      FORMAT (/1X,4X,'ENTER SUBROUTINE RANMAT')
      ENDIF

*     PARAMETERS FOR RANDOM NUMBER GENERATOR.
*     MODULO EQUATION: RANDOM = (RANA*RANDOM + RANC) MODULO RANM.

      RANA   =   671093.0
      RANC   =  7090885.0
      RANM   = 33554432.0

***   {END MODULE 1}
```

```
***** START DO LOOPS FOR RANDOM NUMBER MATRIX RANMTX(IA,IX).  {MODULE 2}

      NIXX = NIX - 1
      DO 140 IA=1,NIA
         DO 140 IX=1,NIXX


*****       CALCULATE OR COPY A SINGLE RANDOM NUMBER RANMTX(IA,IX).
*****       {MODULE 3}

            IF (ITASK(IA,IX) .EQ. 0) THEN


*****          NO TASK VARIABLE TO CONSIDER.  {MODULE 4}

               RANMTX(IA,IX) = 0.0

***            {END MODULE 4}


*****       {CONTINUE MODULE 3}

            ELSE


*****          TASK VARIABLE PRESENT.  {MODULE 5}

*              CALCULATE A NEW RANDOM NUMBER IF RANCAL = .TRUE..

               IF (IA .EQ. 1) THEN

*                 ALWAYS CALCULATE A NEW RANDOM NUMBER FOR IA .EQ. 1.

                  RANCAL = .TRUE.

               ELSE

*                 WILL NEED TO CALCULATE A NEW RANDOM NUMBER IF THE TASK
*                 VARIBLE NOT USED BY PRIOR ALTERNATIVE.  OTHERWISE COPY
*                 THE RANDOM NUMBER.

                  NIAA = IA - 1
                  DO 110 IAA=1,NIAA

                     IF (ITASK(IA,IX) .EQ. ITASK(IAA,IX)) THEN

                        RANMTX(IA,IX) = RANMTX(IAA,IX)
                        RANCAL = .FALSE.
                        GO TO 120

                     ELSE

                        RANCAL = .TRUE.
```

```
                     ENDIF

110                CONTINUE

120                CONTINUE

                ENDIF

***             END TASK VARIABLE PRESENT.  {END MODULE 5}


*****           CALCULATE A SINGLE RANDOM NUMBER.  {MODULE 6}

                IF (RANCAL) THEN

*                  FOR ACCURACY, DO MODULO ARITHMETIC W/O MODULO FUNCTION.
                   RANX           = RANA*RANDOM + RANC
                   RANDIV         = RANX/RANM
                   RANT           = DINT(RANDIV)
                   RANSUB         = RANT*RANM
                   RANDOM         = RANX - RANSUB
                   RANMTX(IA,IX) = SNGL(RANDOM/RANM)

                ENDIF

***             {END MODULE 6}

             ENDIF

***          END CALCULATE A SINGLE RANDOM NUMBER.  {END MODULE 3}

140    CONTINUE

***    END DO LOOPS FOR IA AND IX.  {END MODULE 2}


***** WRITE RANMTX(IA,IX).  {MODULE 7}

       IF (JDIAG .EQ. 2) THEN

          NIXX = NIX - 1
          DO 170 IA=1,NIA

             IF (NIXX .LE. 5) THEN
                WRITE  (*,150) IA,(RANMTX(IA,IX),IX=1,NIXX)
             ELSE
                WRITE  (*,150) IA,(RANMTX(IA,IX),IX=1,5)
                WRITE  (*,160)    (RANMTX(IA,IX),IX=6,NIXX)
             ENDIF

150          FORMAT (1X,I5,3X,5F14.4)
160          FORMAT (1X,   8X,5F14.4)
```

```
170      CONTINUE

      ENDIF

***   {END MODULE 7}


***** EXIT FROM SUBROUTINE RANMAT.  {MODULE 8}

      IF (JDIAG .EQ. 2) THEN
         WRITE (*,999)
999      FORMAT (1X,4X,'EXIT  SUBROUTINE RANMAT')
      ENDIF

      RETURN

      END

***   {END MODULE 8}

****************************** RANMAT.FOR **********************************
```

File: VARCAL.FOR

```
************************************************************************
*                          VARCAL.FOR                                  *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.5.3.3                                                    *
*  DATE:    01/29/85                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE VARCAL.  IT CALCULATES THE RANDOM VARIABLE   *
*  MATRIX FOR TASK VARIABLES FROM THE RANDOM NUMBER MATRIX USING THE   *
*  CDF COEFFICIENTS GENERATED BY THE SUBROUTINE COEFF.                 *
*                                                                      *
*----------------------------------------------------------------------*
*                     CONFIGURATION CHANGES                            *
*                                                                      *
*    DATE                    CHANGE                                    *
*                                                                      *
*  01/18/85  * ORIGINAL.                                               *
*  01/22/85  * MODULE  8: ID LOOP CHANGED TO IC LOOP FOR CLARITY.      *
*  01/29/85  * MODULE  1: JDIAG .EQ. 2.                                *
*            * MODULE 11: JDIAG .EQ. 2.                                *
*                                                                      *
************************************************************************

$TITLE:'VARCAL.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE VARCAL

************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .EQ. 2) THEN
        WRITE  (*,100)
100     FORMAT (/1X,4X,'ENTER SUBROUTINE VARCAL')
      ENDIF

***   {END MODULE 1}


***** START (IA,IX) DO LOOPS FOR TASK VARIABLES VARMTX(IA,IX).
***** {MODULE 2}

      NIXX = NIX - 1
      DO 130 IA=1,NIA
        DO 130 IX=1,NIXX
```

```
            IDTASK = ITASK(IA,IX)
            RAN    = RANMTX(IA,IX)


*****       TEST WHETHER TASK VARIABLE USED AND MAKE TASK VARIABLE
*****       ASSIGNMENT.  {MODULE 3}

            IF (IDTASK .EQ. 0) THEN


*****          SET UNUSED TASK VARIABLE VARMTX(IA,IX) = 0.0.  {MODULE 4}

               VARMTX(IA,IX) = 0.0

***            {END MODULE 4}


*****       {CONTINUE MODULE 3}

            ELSE

*****          TASK VARIABLE VARMTX(IA,IX) USED.  TEST FOR SUCCESS OF
*****          TASK AND MAKE TASK VARIABLE ASSIGNMENT.  {MODULE 5}

               IF (RAN .GT. TDATA(IDTASK,2)) THEN


*****             TASK FAILED.  SET VARMTX(IA,IX) TO TASK FAILURE VALUE.
*****             {MODULE 6}

                  VARMTX(IA,IX) = TDATA(IDTASK,1)

***               {END MODULE 6}


*****          {CONTINUE MODULE 5}

               ELSE


*****             TASK SUCCESSFUL. CALCULATE CDF CONDITIONAL ON SUCCESS.
*****             {MODULE 7}

                  RAN1 = RAN/TDATA(IDTASK,2)

*                 ADJUST RAN1 IF > 1.0 DUE TO ROUNDOFF.

                  IF (RAN1 .GT. 1.0) RAN1 = 1.0

***               {END MODULE 7}


*****             START DO LOOP TO CALCULATE VARMTX(IA,IX) FOR
*****             SUCCESSFUL TASK.  {MODULE 8}
```

```
                  DO 110 IC=1,NIC,2

*****                FIND CORRECT TASK VARIABLE COEFFICIENTS AND
*****                CALCULATE VARMTX(IA,IX).  {MODULE 9}

                     ID = IC + 4

                     IF (RAN1 .LE. TDATA(IDTASK,ID+1)) THEN

*                       CORRECT TASK VARIABLE COEFFICIENTS.  CALCULATE
*                       VARMTX(IA,IX).

                        VARMTX(IA,IX) = TCOEF(IDTASK,IC)*RAN1 +
     *                                  TCOEF(IDTASK,IC+1)

*                       JUMP OUT OF (IC) DO LOOP FOR CALCULATING ONE
*                       SUCCESSFUL TASK VARIABLE VARMTX(IA,IX).

                        GO TO 120

                     ENDIF

***                  {END MODULE 9}

110                  CONTINUE

120                  CONTINUE

***                  END (IC) DO LOOP TO CALCULATE VARMTX(IA,IX) FOR ONE
***                  SUCCESSFUL TASK VARIABLE.  {END MODULE 8}

               ENDIF

***            END ONE TASK VARIABLE VARMTX(IA,IX) USED AND TASK
***            VARIABLE ASSIGNMENT.  {END MODULE 5}

            ENDIF

***         END TEST WHETHER ONE TASK VARIABLE USED AND ONE TASK
***         VARIABLE ASSIGNMENT.  {END MODULE 3}

130   CONTINUE

***   END (IA,IX) DO LOOPS FOR TASK VARIABLES VARMTX(IA,IX).
***   {END MODULE 2}


***** WRITE TASK VARIABLES VARMTX(IA,IX).  {MODULE 10}

      IF (JDIAG .EQ. 2) THEN

         NIXX = NIX - 1
```

```
         DO 160 IA=1,NIA

            IF (NIXX .LE. 5) THEN
               WRITE  (*,140) IA,(VARMTX(IA,IX),IX=1,NIXX)
            ELSE
               WRITE  (*,140) IA,(VARMTX(IA,IX),IX=1,5)
               WRITE  (*,150)    (VARMTX(IA,IX),IX=6,NIXX)
            ENDIF

140         FORMAT (1X,I5,3X,1P5E14.4)
150         FORMAT (1X,   8X,1P5E14.4)

160      CONTINUE

      ENDIF

***   {END MODULE 10}


***** EXIT FROM SUBROUTINE VARCAL.  {MODULE 11}

      IF (JDIAG .EQ. 2) THEN
         WRITE (*,999)
999      FORMAT (1X,4X,'EXIT  SUBROUTINE VARCAL')
      ENDIF

      RETURN

      END

***   {END MODULE 11}

****************************** VARCAL.FOR ********************************
```

File: PRICE.FOR

```
************************************************************************
*                              PRICE.FOR                               *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.5.3.4                                                    *
*  DATE:    02/04/85                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE PRICE.  IT CALCULATES THE TASK PRICES AND    *
*  THE TOTAL PRICES FOR A SINGLE MONTE CARLO TRIAL USING THE RANDOM    *
*  VARIABLE VALUES GENERATED BY THE SUBROUTINE VARCAL.  IT DETERMINES  *
*  THE MINIMUM TOTAL PRICE AND THE ASSOCIATED STEP PRICES FOR A        *
*  TRIAL, AND INCREMENTS THE MINIMUM ALTERNATIVE HISTOGRAM.  IT        *
*  CALCULATES UTILITY FUNCTION VALUES FOR THE MINIMUM TOTAL PRICE.     *
*                                                                      *
*----------------------------------------------------------------------*
*                     CONFIGURATION CHANGES                            *
*                                                                      *
*    DATE                    CHANGE                                    *
*                                                                      *
*  12/05/84  * ORIGINAL.                                               *
*  02/04/85  * MODULE  1: JDIAG .EQ. 2.                                *
*            * MODULE 10: JDIAG .EQ. 2.                                *
*                         ADD: IF ((JDIAG .EQ. 1 .AND. ) ...) ETC.     *
*            * MODULE 13: JDIAG .EQ. 2.                                *
*                                                                      *
************************************************************************

$TITLE:'PRICE.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE PRICE

************************************************************************


***** INITIALIZE.  {MODULE 1}

      LOGICAL INALT,UCAL

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .EQ. 2) THEN
         WRITE  (*,100)
100      FORMAT (/1X,4X,'ENTER SUBROUTINE PRICE')
      ENDIF

      INALT = .FALSE.

***   {END MODULE 1}
```

File: PRICE.FOR

```
***** START (IA) DO LOOP TO CALCULATE PRICES.  {MODULE 2}

      DO 130 IA=1,NIA

*****    IF ALTERNATIVE IA NOT TO BE INCLUDED IN RUN (INCLUD .EQ. 0),
*****    SET PMTX(IA,IP) = 0.0, ELSE CALCULATE STEP AND TOTAL PRICES.
*****    {MODULE 3}

         INCLUD = ITASK(IA,NIX)

         IF (INCLUD .EQ. 0) THEN

*****       ALTERNATIVE IA NOT INCLUDED IN RUN (INCLUD .EQ. 0).
*****       {MODULE 4}

            DO 110 IP=1,NIP
               PMTX(IA,IP) = 0.0
110         CONTINUE

***         {END MODULE 4}

*****    {CONTINUE MODULE 3}

         ELSE

*****       ALTERNATIVE IA INCLUDED IN RUN (INCLUD .EQ. 1).
*****       CALCULATE STEP AND TOTAL PRICES PMTX(IA,IP).
*****       {MODULES 4-5}

$INCLUDE:'EQNS.FOR'

***         {END MODULES 4-5}

*****       IF NO ALTERNATIVE YET INCLUDED IN TRIAL OR THIS ALTERNATIVE
*****       HAS THE MINIMUM TOTAL PRICE FOR THE RUN,
*****       PMIN(IP) = PMTX(IA,IP).  {MODULE 6}

            IF ((.NOT. INALT) .OR. (PMTX(IA,NIP) .LE. PMIN(NIP))) THEN

*              SET PMIN(IP) EQUAL TO PRICES OF THIS ALTERNATIVE.

               DO 120 IP=1,NIP
                  PMIN(IP) = PMTX(IA,IP)
120            CONTINUE

               IAMIN = IA
               INALT = .TRUE.
```

```
            ENDIF
***         {END MODULE 6}
         ENDIF
***      END IF (INCLUD .EQ. 0).  {END MODULE 3}
130   CONTINUE
***   END (IA) DO LOOP TO CALCULATE PRICES.  {END MODULE 2}

***** CALCULATE THE UTILITY FUNCTION VALUES UTIL(II) FOR PMIN(NIP).
***** {MODULE 7}
      DO 150 II=1,NII
         IF (PMIN(NIP) .LE. UDATA(II,1)) THEN
            UTIL(II) = 1.0
         ELSE
            UCAL = .FALSE.
            DO 140 ID=1,NIU,2
               IF (UDATA(II,ID+1) .EQ. 0.0) THEN
                  UTIL(II) = 0.0
                  UCAL     = .TRUE.
               ELSEIF (PMIN(NIP) .LE. UDATA(II,ID+2)) THEN
                  UTIL(II) = UCOEF(II,ID)*PMIN(NIP) + UCOEF(II,ID+1)
                  UCAL     = .TRUE.
               ENDIF
               IF (UCAL) GO TO 150
140         CONTINUE
         ENDIF
150   CONTINUE
***   {END MODULE 7}

***** INCREMENT THE MINIMUM ALTERNATIVE HISTOGRAM.
***** {MODULE 8}
```

```
      IHALT(IAMIN) = IHALT(IAMIN) + 1

***   {END MODULE 8}


***** WRITE PMTX(IA,IP).  {MODULE 9}

      IF (JDIAG .EQ. 2) THEN

         WRITE  (*,*) 'PRICE MATRIX PMTX(IA,IP) FOR THIS TRIAL'

         DO 180 IA=1,NIA

            IF (NIP .LE. 5) THEN
               WRITE (*,160) IA,(PMTX(IA,IP),IP=1,NIP)
            ELSE
               WRITE (*,160) IA,(PMTX(IA,IP),IP=1,5)
               WRITE (*,170)    (PMTX(IA,IP),IP=6,NIP)
            ENDIF

160         FORMAT (1X,I5,3X,1P5E14.4)
170         FORMAT (1X,   8X,1P5E14.4)

180      CONTINUE

      ENDIF

***   {END MODULE 9}


***** WRITE PMIN(IP).  {MODULE 10}

      IF ((JDIAG .EQ. 1) .AND. (MOD(IITR,100) .EQ. 0))
     *   WRITE (*,190) PMIN(NIP)

      IF (JDIAG .EQ. 2) THEN

         WRITE  (*,*) 'MINIMUM PRICES FOR THIS TRIAL'

         IF (IP .LE. 5) THEN
            WRITE  (*,190) (PMIN(IP),IP=1,5)
         ELSE
            WRITE  (*,190) (PMIN(IP),IP=1,5)
            WRITE  (*,200) (PMIN(IP),IP=6,NIP)
         ENDIF

190      FORMAT (1X,'PMIN',4X,1P5E14.4)
200      FORMAT (1X,       8X,1P5E14.4)

      ENDIF

***   {END MODULE 10}
```

File: PRICE.FOR

```
***** WRITE UTIL(II).  {MODULE 11}

      IF (JDIAG .EQ. 2) THEN

         WRITE  (*,*) 'UTILITY FUNCTION VALUES FOR THIS TRIAL'

         IF (NII .LE. 5) THEN
            WRITE  (*,210) (UTIL(II),II=1,NII)
         ELSE
            WRITE  (*,210) (UTIL(II),II=1,5)
            WRITE  (*,220) (UTIL(II),II=6,NII)
         ENDIF

      ENDIF

210   FORMAT (1X,'UTIL',4X,5F14.4)
220   FORMAT (1X,       8X,5F14.4)

***   {END MODULE 11}


***** WRITE IHALT(IA).  {MODULE 12}

      IF (JDIAG .EQ. 2) THEN

         WRITE  (*,*) 'ALTERNATIVE HISTOGRAM IHALT(IA)'

         IF (NIA .LE. 10) THEN
            WRITE  (*,230) (IHALT(IA),IA=1,NIA)
         ELSE
            WRITE  (*,230) (IHALT(IA),IA=1,10)
            WRITE  (*,240) (IHALT(IA),IA=11,NIA)
         ENDIF

230      FORMAT (1X,'IHALT',3X,10I5)
240      FORMAT (1X,        8X,10I5)

      ENDIF

***   {END MODULE 12}


***** EXIT FROM SUBROUTINE PRICE.  {MODULE 13}

      IF (JDIAG .EQ. 2) THEN
         WRITE  (*,999)
999      FORMAT (1X,4X,'EXIT  SUBROUTINE PRICE')
      ENDIF

      RETURN

      END

***   {END MODULE 13}
```

File: PRICE.FOR

```
****************************** PRICE.FOR ********************************
```

File: RUNHIS.FOR

```
************************************************************************
*                          RUNHIS.FOR                                  *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.5.4                                                      *
*  DATE:    01/29/85                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE RUNHIS.  IT GENERATES A HISTOGRAM OF         *
*  PRICES.                                                             *
*                                                                      *
*----------------------------------------------------------------------*
*                     CONFIGURATION CHANGES                            *
*                                                                      *
*    DATE                    CHANGE                                    *
*                                                                      *
*  12/04/84  * ORIGINAL.                                               *
*  01/29/85  * MODULE  1: JDIAG .EQ. 2                                 *
*              MODULE  8: JDIAG .EQ. 2                                 *
*                                                                      *
************************************************************************

$TITLE:'RUNHIS.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE RUNHIS

************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .EQ. 2) THEN
         WRITE  (*,100)
100      FORMAT (/1X,'  ENTER SUBROUTINE RUNHIS')
      ENDIF

***   {END MODULE 1}


***** START (IP) DO LOOP WITH PRICES PMIN(IP) TO INCREMENT HISTOGRAMS.
***** {MODULE 2}

      DO 120 IP=1,NIP


*****    CALCULATE HISTOGRAM INTERVAL DEL = 1/50 OF RANGE.  {MODULE 3}

         DEL = (PUB(IP) - PLB(IP))/50
```

```
***       {END MODULE 3}


*****     INCREMENT HISTOGRAM IF PMIN(IP) BELOW RANGE OR EQUAL TO
*****     PLB(IP).  {MODULE 4}

          IF (PMIN(IP) .LE. PLB(IP)) THEN

             IH = 1
             IHPMIN(IP,IH) = IHPMIN(IP,IH) + 1

***       {END MODULE 4}


*****     INCREMENT HISTOGRAM IF PMIN(IP) IN RANGE.  {MODULE 5}

          ELSEIF (PMIN(IP) .LE. PUB(IP)) THEN

             IF (PMIN(IP) - PLB(IP) .EQ. FLOAT(INT(PMIN(IP) - PLB(IP))))
     *       THEN
                IH = INT((PMIN(IP) - PLB(IP))/DEL) + 1
             ELSE
                IH = INT((PMIN(IP) - PLB(IP))/DEL) + 2
             ENDIF

             IHPMIN(IP,IH) = IHPMIN(IP,IH) + 1

***       {END MODULE 5}


*****     INCREMENT HISTOGRAM IF PMIN(IP) ABOVE RANGE.  {MODULE 6}

          ELSE

             IH = 52
             IHPMIN(IP,IH) = IHPMIN(IP,IH) + 1

          ENDIF

***       {END MODULE 6}


*****     WRITE HISTOGRAM BIN.  {MODULE 7}

          IF (JDIAG .EQ. 2) THEN
             WRITE  (*,110) IP,IH,PMIN(IP)
110          FORMAT (1X,'IP = ',I5,12X,'IH = ',I5,12X,
     *               'PMIN(IP) = ',1PE14.4)
          ENDIF

***       {END MODULE 7}

120    CONTINUE
```

File: RUNHIS.FOR

```
***   END (IP) DO LOOP.  {End Module 2}

***** EXIT FROM SUBROUTINE RUNHIS.  {MODULE 8}

      IF (JDIAG .EQ. 2) THEN
         WRITE  (*,999)
999      FORMAT (1X,'  EXIT  SUBROUTINE RUNHIS')
      ENDIF

      RETURN

      END

***   {END MODULE 8}

****************************** RUNHIS.FOR **********************************
```

```
************************************************************************
*                              RUNSTA.FOR                              *
*                                                                      *
*  PROGRAM: SIMRAND I                                                  *
*  MODULE:  1.5.5                                                      *
*  DATE:    02/04/85                                                   *
*                                                                      *
*  THIS IS THE SUBROUTINE RUNSTA.  IT ACCUMULATES THE STATISTICS       *
*  FOR THE RUN.                                                        *
*                                                                      *
*----------------------------------------------------------------------*
*                      CONFIGURATION CHANGES                           *
*                                                                      *
*    DATE                    CHANGE                                    *
*                                                                      *
*  01/18/85  * ORIGINAL.                                               *
*  01/22/85  * MODULE 13: REPLACE PRMIN WITH PRMAX.                    *
*  01/29/85  * MODULE  1: JDIAG .EQ. 2                                 *
*            * MODULE 17: JDIAG .EQ. 2                                 *
*  02/04/85  * MODULE 10: IF ((JDIAG .EQ. 1) .AND. (MOD(IITR,100) ...  *
*            * MODULE 16: IF ((JDIAG .EQ. 1 .AND. MOD(IITR,100)) ...   *
*                                                                      *
************************************************************************

$TITLE:'RUNSTA.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

************************************************************************

      SUBROUTINE RUNSTA

************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .EQ. 2) THEN
         WRITE (*,100)
100      FORMAT (/1X,'  ENTER SUBROUTINE RUNSTA')
      ENDIF

***   {END MODULE 1}


***** START DO LOOP TO CALCULATE RUN STATISTICS.  {MODULE 2}

      DO 110 IP=1,NIP


*****    CALCULATE PMEAN(IP).  {MODULE 3}
```

```
         PMEAN(IP) = ((IITR-1)*PMEAN(IP) + PMIN(IP))/IITR

***      {END MODULE 3}


*****    CALCULATE PMIN2(IP).  {MODULE 4}

         PMIN2(IP) = PMIN2(IP) + PMIN(IP)**2

***      {END MODULE 4}


*****    CALCULATE PSDEV(IP).  {MODULE 5}

         IF (IITR .GT. 1) THEN
            PVAR(IP) = (PMIN2(IP) - IITR*PMEAN(IP)**2)/(IITR - 1)
         ENDIF

         PSDEV(IP) = SQRT(ABS(PVAR(IP)))

***      {END MODULE 5}


*****    DETERMINE MINIMUM TOTAL PRICE & MINIMUM STEP PRICES.
*****    {MODULE 6}

         IF ((IITR .EQ. 1) .OR. (PMIN(IP) .LT. PRMIN(IP))) THEN
            PRMIN(IP) = PMIN(IP)
         ENDIF

***      {END MODULE 6}


*****    DETERMINE MAXIMUM TOTAL PRICE & MAXIMUM STEP PRICES.
*****    {MODULE 7}

         IF ((IITR .EQ. 1) .OR. (PMIN(IP) .GT. PRMAX(IP))) THEN
            PRMAX(IP) = PMIN(IP)
         ENDIF

***      {END MODULE 7}

110   CONTINUE

***   END IP DO LOOP.  {MODULE 2}


***** CALCULATE RUN UTILITY FUNCTION VALUES.  {MODULE 8}

      DO 120 II=1,NII
         RUTIL(II) = ((IITR - 1)*RUTIL(II) + UTIL(II))/IITR
120   CONTINUE
```

```
***   {END MODULE 8}


***** CALCULATE CERTAINTY EQUIVALENTS.  {MODULE 9}

      DO 150 II=1,NII

         IF (RUTIL(II) .LT. 0.0) RUTIL(II) = 0.0
         IF (RUTIL(II) .GT. 1.0) RUTIL(II) = 1.0

         DO 130 ID=1,NID,2
            IID = ID
            IF (RUTIL(II) .GE. UDATA(II,ID+3)) GO TO 140
130      CONTINUE

140      CERTEQ(II) = CECOEF(II,IID)*RUTIL(II) + CECOEF(II,IID+1)

150   CONTINUE

***   {END MODULE 9}


***** WRITE PMEAN(IP).  {MODULE 10}

      IF ((JDIAG .EQ. 1) .AND. (MOD(IITR,100) .EQ. 0))
     *   WRITE  (*,160) PMEAN(NIP)

      IF (JDIAG .EQ. 2) THEN

         IF (NIP .LE. 5) THEN
            WRITE  (*,160) (PMEAN(IP),IP=1,NIP)
         ELSE
            WRITE  (*,160) (PMEAN(IP),IP=1,5)
            WRITE  (*,170) (PMEAN(IP),IP=6,NIP)
         ENDIF

160      FORMAT (1X,'PMEAN',3X,1P5E14.4)
170      FORMAT (1X,       8X,1P5E14.4)

      ENDIF

***   {END MODULE 10}


***** WRITE PSDEV(IP).  {MODULE 11}

      IF (JDIAG .EQ. 2) THEN

         IF (NIP .LE. 5) THEN
            WRITE  (*,180) (PSDEV(IP),IP=1,NIP)
         ELSE
            WRITE  (*,180) (PSDEV(IP),IP=1,5)
            WRITE  (*,190) (PSDEV(IP),IP=6,NIP)
         ENDIF
```

```
180      FORMAT (1X,'PSDEV',3X,1P5E14.4)
190      FORMAT (1X,        8X,1P5E14.4)

      ENDIF

***   {END MODULE 11}


***** WRITE PRMIN(IP).  {MODULE 12}

      IF (JDIAG .EQ. 2) THEN

         IF (NIP .LE. 5) THEN
            WRITE  (*,200) (PRMIN(IP),IP=1,5)
         ELSE
            WRITE  (*,200) (PRMIN(IP),IP=1,5)
            WRITE  (*,210) (PRMIN(IP),IP=6,NIP)
         ENDIF

200      FORMAT (1X,'PRMIN',3X,1P5E14.4)
210      FORMAT (1X,        8X,1P5E14.4)

      ENDIF

***   {END MODULE 12}


***** WRITE PRMAX(IP).  {MODULE 13}

      IF (JDIAG .EQ. 2) THEN

         IF (NIP .LE. 5) THEN
            WRITE  (*,220) (PRMAX(IP),IP=1,5)
         ELSE
            WRITE  (*,220) (PRMAX(IP),IP=1,5)
            WRITE  (*,230) (PRMAX(IP),IP=6,NIP)
         ENDIF

220      FORMAT (1X,'PRMAX',3X,1P5E14.4)
230      FORMAT (1X,        8X,1P5E14.4)

      ENDIF

***   {END MODULE 13}


***** WRITE RUTIL(II).  {MODULE 14}

      IF (JDIAG .EQ. 2) THEN

         IF (NII .LE. 5) THEN
            WRITE  (*,240) (RUTIL(II),II=1,NII)
         ELSE
```

```
            WRITE  (*,240) (RUTIL(II),II=1,5)
            WRITE  (*,250) (RUTIL(II),II=6,NII)
         ENDIF

240      FORMAT (1X,'RUTIL',3X,5F14.4)
250      FORMAT (1X,        8X,5F14.4)

      ENDIF

***   {END MODULE 14}


***** WRITE CERTEQ(II).  {MODULE 15}

      IF (JDIAG .EQ. 2) THEN

         IF (NII .LE. 5) THEN
            WRITE  (*,260) (CERTEQ(II),II=1,NII)
         ELSE
            WRITE  (*,260) (CERTEQ(II),II=1,5)
            WRITE  (*,270) (CERTEQ(II),II=6,NII)
         ENDIF

260      FORMAT (1X,'CERTEQ',2X,5F14.4)
270      FORMAT (1X,         8X,5F14.4)

      ENDIF

***   {END MODULE 15}


***** WRITE IHALT(IA).  {MODULE 16}

     IF ((JDIAG .EQ. 1 .AND. MOD(IITR,100) .EQ. 0) .OR.
    *    (JDIAG .EQ. 2)) THEN

         IF (NIA .LE. 10) THEN
            WRITE (*,280) (IHALT(IA),IA=1,NIA)
         ELSE
            WRITE (*,280) (IHALT(IA),IA=1,10)
            WRITE (*,290) (IHALT(IA),IA=11,NIA)
         ENDIF

280      FORMAT (1X,'IHALT',3X,10I5)
290      FORMAT (1X,        8X,10I5)

      ENDIF

***   {END MODULE 16}


***** EXIT FROM SUBROUTINE RUNSTA.  {MODULE 17}

      IF (JDIAG .EQ. 2) THEN
```

```
         WRITE (*,999)
999      FORMAT (1X,'  EXIT  SUBROUTINE RUNSTA')
      ENDIF

      RETURN

      END

***   {END MODULE 17}
```

File: OUTPUT.FOR

```
*************************************************************************
*                            OUTPUT.FOR                                 *
*                                                                       *
*  PROGRAM: SIMRAND I                                                   *
*  MODULE:  1.6                                                         *
*  DATE:    04/09/85                                                    *
*                                                                       *
*  THIS IS THE SUBROUTINE OUTPUT.  IT CALCULATES THE TOTAL STATISTICS   *
*  FOR THE RUN AND OUTPUTS THE RESULTS TO A FILE.                       *
*                                                                       *
*-----------------------------------------------------------------------*
*                      CONFIGURATION CHANGES                            *
*                                                                       *
*    DATE                    CHANGE                                     *
*                                                                       *
*  01/18/85  * ORIGINAL.                                                *
*  04/09/85  * MODULE 10: REPLACE PRMIN WITH PRMAX.                     *
*                                                                       *
*************************************************************************

$TITLE:'OUTPUT.LST'
$DEBUG
$NOFLOATCALLS
$STORAGE:2

*************************************************************************

      SUBROUTINE OUTPUT

*************************************************************************


***** INITIALIZE.  {MODULE 1}

$INCLUDE:'INITIAL.FOR'

      IF (JDIAG .NE. 0) THEN
         WRITE  (*,100)
100      FORMAT (/1X,'ENTER SUBROUTINE OUTPUT')
      ENDIF

*     PROBABILITY POINTS FOR SUMMARY CDFS.
      ZPCT(1) = 0.10
      ZPCT(2) = 0.25
      ZPCT(3) = 0.50
      ZPCT(4) = 0.75
      ZPCT(5) = 0.90

***   {END MODULE 1}


***** START DO LOOP TO CALCULATE CDFS.  {MODULE 2}

      DO 150 IP=1,NIP
```

```
*****    CALCULATE PRICEZ(IP,IZ) FOR SUMMARY CDFS.  {MODULE 3}

         DO 130 IZ=1,5

            PERCNT = FLOAT(IHPMIN(IP,1))/(NTRIAL)

            IF (PERCNT .GE. ZPCT(IZ)) THEN

               PRICEZ(IP,IZ) = PLB(IP)

            ELSEIF (PERCNT .LT. 1.00) THEN

               DO 110 IH=2,51
                  IIH = IH
                  PERCNT = PERCNT + FLOAT(IHPMIN(IP,IH))/(NTRIAL)
                  IF (PERCNT .GE. ZPCT(IZ)) GO TO 120
110            CONTINUE

120            CONTINUE

               PRICEZ(IP,IZ) = PLB(IP) +
     *                         (IIH - 1)*(PUB(IP) - PLB(IP))/50

            ELSE

               PRICEZ(IP,IZ) = PUB(IP)

            ENDIF

130      CONTINUE

***      {END MODULE 3}


*****    CALCULATE PRICE CDF(IP,IH) & PCDF(IP,IH) FOR DETAILED CDFS.
*****    {MODULE 4}

         CDF(IP,1)   = FLOAT(IHPMIN(IP,1))/(NTRIAL)
         PCDF(IP,1)  = PLB(IP)

         DO 140 IH=2,51
            CDF(IP,IH)  = CDF(IP,IH-1) + FLOAT(IHPMIN(IP,IH))/(NTRIAL)
            PCDF(IP,IH) = PLB(IP) + (IH - 1)*(PUB(IP) - PLB(IP))/50
140      CONTINUE

***      {END MODULE 4}

150   CONTINUE

***   {END MODULE 2}
```

```
***** WRITE RUN IDENTIFICATION TO FILE.  {MODULE 5}

      WRITE  (2,160) IDATA,NTRIAL,RNSEED
160   FORMAT ('IDATA = ',I5,8X,'NTRIAL = ',I5,8X,'RANDOM SEED = ',F15.0)

***   {END MODULE 5}


***** WRITE PMEAN(IP).  {MODULE 6}

      IF (NIP .LE. 5) THEN
         WRITE  (2,170) (PMEAN(IP),IP=1,NIP)
      ELSE
         WRITE  (2,170) (PMEAN(IP),IP=1,5)
         WRITE  (2,180) (PMEAN(IP),IP=6,NIP)
      ENDIF

170   FORMAT ('PMEAN',3X,1P5E14.4)
180   FORMAT (        8X,1P5E14.4)

***   {END MODULE 6}


***** WRITE PSDEV(IP).  {MODULE 7}

      IF (NIP .LE. 5) THEN
         WRITE  (2,190) (PSDEV(IP),IP=1,NIP)
      ELSE
         WRITE  (2,190) (PSDEV(IP),IP=1,5)
         WRITE  (2,200) (PSDEV(IP),IP=6,NIP)
      ENDIF

190   FORMAT ('PSDEV',3X,1P5E14.4)
200   FORMAT (        8X,1P5E14.4)

***   {END MODULE 7}


***** WRITE PRMIN(IP).  {MODULE 8}

      IF (NIP .LE. 5) THEN
         WRITE  (2,210) (PRMIN(IP),IP=1,5)
      ELSE
         WRITE  (2,210) (PRMIN(IP),IP=1,5)
         WRITE  (2,220) (PRMIN(IP),IP=6,NIP)
      ENDIF

210   FORMAT ('PRMIN',3X,1P5E14.4)
220   FORMAT (        8X,1P5E14.4)

***   {END MODULE 8}


***** WRITE PRICEZ(IP,IZ) TO FILE.  {MODULE 9}
```

```
      DO 250 IZ=1,5

         IF (NIP .LE. 5) THEN
            WRITE  (2,230) ZPCT(IZ),(PRICEZ(IP,IZ),IP=1,NIP)
         ELSE
            WRITE  (2,230) ZPCT(IZ),(PRICEZ(IP,IZ),IP=1,5)
            WRITE  (2,240)          (PRICEZ(IP,IZ),IP=6,NIP)
         ENDIF

230      FORMAT (F4.2,4X,1P5E14.4)
240      FORMAT (     8X,1P5E14.4)

250   CONTINUE

***   {END MODULE 9}


***** WRITE PRMAX(IP).  {MODULE 10}

      IF (NIP .LE. 5) THEN
         WRITE  (2,260) (PRMAX(IP),IP=1,5)
      ELSE
         WRITE  (2,260) (PRMAX(IP),IP=1,5)
         WRITE  (2,270) (PRMAX(IP),IP=6,NIP)
      ENDIF

260   FORMAT ('PRMAX',3X,1P5E14.4)
270   FORMAT (        8X,1P5E14.4)

***   {END MODULE 10}


***** WRITE RUTIL(II).  {MODULE 11}

      IF (NII .LE. 5) THEN
         WRITE  (2,280) (RUTIL(II),II=1,NII)
      ELSE
         WRITE  (2,280) (RUTIL(II),II=1,5)
         WRITE  (2,290) (RUTIL(II),II=6,NII)
      ENDIF

280   FORMAT ('RUTIL',3X,5F14.4)
290   FORMAT (        8X,5F14.4)

***   {END MODULE 11}


***** WRITE CERTEQ(II).  {MODULE 12}

      IF (NII .LE. 5) THEN
         WRITE  (2,300) (CERTEQ(II),II=1,NII)
      ELSE
         WRITE  (2,300) (CERTEQ(II),II=1,5)
```

```
         WRITE  (2,310) (CERTEQ(II),II=6,NII)
      ENDIF

300   FORMAT ('CERTEQ',2X,5F14.4)
310   FORMAT (         8X,5F14.4)

***   {END MODULE 12}


***** WRITE IHALT(IA).  {MODULE 13}

      IF (NIA .LE. 10) THEN
         WRITE (2,320) (IHALT(IA),IA=1,NIA)
      ELSE
         WRITE (2,320) (IHALT(IA),IA=1,10)
         WRITE (2,330) (IHALT(IA),IA=11,NIA)
      ENDIF

320   FORMAT ('IHALT',3X,10I5)
330   FORMAT (        8X,10I5)

***   {END MODULE 13}


***** WRITE RANDOM TO FILE.  {MODULE 14}

*     THIS VALUE OF RANDOM IS THE LAST GENERATED AND SHOULD BE USED AS
*     THE SEED (RNSEED) TO REPEAT THE RUN WITH DIFFERENT RANDOM NUMBERS.

      WRITE  (2,340) RANDOM
340   FORMAT ('RANDOM NUMBER SEED (RNSEED) FOR NEXT RUN = ',F15.0)

***   {END MODULE 14}


***** IF JLOUT = 1, WRITE DETAILED CDFS TO OUTPUT FILE.  {MODULE 15}

      IF (JLOUT .EQ. 1) THEN


*****    WRITE IH, STEP PRICE, & CDF(IP,IH) TO FILE.  {MODULE 16}

         DO 410 IP=1,NIP

            WRITE  (2,350) IP
350         FORMAT (/14X,'IH, STEP PRICE, & CDF(IP,IH) FOR IP = ',I5)

            WRITE  (2,360) PRMIN(IP),PRMAX(IP)
360         FORMAT (/6X,'PRMIN(IP) = ',1PE14.4,5X,'PRMAX(IP) = ',
     *              1PE14.4)

            WRITE  (2,370)
370         FORMAT (/'   IH ','     PRICE     ','   CDF(IP,IH)',
     *              3X,'   IH ','     PRICE     ','   CDF(IP,IH)'/)
```

```
            DO 390 IH=1,25

               IH25 = IH + 25

               WRITE (2,380) IH,  PCDF(IP,IH),  CDF(IP,IH),
     *                        IH25,PCDF(IP,IH25),CDF(IP,IH25)
380            FORMAT (2(I5,1PE14.4,0PF14.4,4X))

390         CONTINUE

            WRITE  (2,400) PCDF(IP,51),CDF(IP,51)
400         FORMAT (40X,'51',1PE14.4,0PF14.4)

410      CONTINUE

***      {END MODULE 16}


*****    WRITE IHPMIN(IP,IH) TO FILE.  {MODULE 17}

         DO 440 IP=1,NIP

            WRITE  (2,420) IP,(IHPMIN(IP,IH),IH=1,10)
            WRITE  (2,430)    (IHPMIN(IP,IH),IH=11,52)

420         FORMAT ('IHPMIN',4X,I5, 5X,10I5)
430         FORMAT (                20X,10I5)

440      CONTINUE

***      {END MODULE 17}

      ENDIF

***   {END MODULE 15}


***** EXIT FROM SUBROUTINE OUTPUT.  {MODULE 18}

      IF (JDIAG .NE. 0) THEN
         WRITE  (*,999)
999      FORMAT (1X,'EXIT  SUBROUTINE OUTPUT')
      ENDIF

      RETURN

      END

***   {END MODULE 18}

****************************** OUTPUT.FOR ********************************
```

File: INITIAL.FOR

```
*************************************************************************
*                              INITIAL.FOR                              *
*                                                                       *
*  PROGRAM: SIMRAND I                                                   *
*  DATE:    12/05/84                                                    *
*                                                                       *
*  THIS SIMRAND I MODULE IS 'INCLUDED' IN THE MAIN ROUTINE AND ALL      *
*  SUBROUTINES AND IS ARRAYED FOR THE LARGE SCALE SYSTEMS ARTICLE.      *
*  IT TYPES THE COMMON VARIABLES.  IT DIMENSIONS THE ARRAYS AND         *
*  DEFINES THE COMMON BLOCKS.  IT EQUIVALENCES TWO ARRAYS.              *
*                                                                       *
*-----------------------------------------------------------------------*
*                         CONFIGURATION CHANGES                         *
*                                                                       *
*    DATE                      CHANGE                                   *
*                                                                       *
*  12/05/84   * ORIGINAL.                                               *
*                                                                       *
*************************************************************************


***** TYPE COMMON VARIABLES.  {MODULE 1}

      DOUBLE PRECISION RNSEED,RANDOM

***   {END MODULE 1}


***** SIZE ARRAYS AS COMMENTED.  (NIXX = NIX - 1) (NIC = NID - 4)
***** (NIU = NID - 2)  {MODULE 2}

*     DIMENSION
*    *   ITASK(NIA,NIX),TDATA(NIT,NID),TCOEF(NIT,NIC),
*    *   RANMTX(NIA,NIXX),VARMTX(NIA,NIXX),X(NIXX),
*    *   UDATA(NII,NIU),UCOEF(NII,NIC),CECOEF(NII,NIC),
*    *   PLB(NIP),PUB(NIP),
*    *   PMEAN(NIP),PMIN2(NIP),PVAR(NIP),PSDEV(NIP),
*    *   IHALT(NIA),IHPMIN(NIP,52),
*    *   UTIL(NII),RUTIL(NII),CERTEQ(NII),
*    *   PMIN(NIP),PRMIN(NIP),PRMAX(NIP),PMTX(NIA,NIP),
*    *   PRICEZ(NIP,5),CDF(NIP,52),PCDF(NIP,52),ZPCT(5)

***   {END MODULE 2}


***** SIZE ARRAYS.  {MODULE 3}

      DIMENSION
     *   ITASK(4,7),TDATA(10,16),TCOEF(10,12),
     *   RANMTX(4,6),VARMTX(4,6),X(6),
     *   UDATA(1,14),UCOEF(1,12),CECOEF(1,12),
     *   PLB(5),PUB(5),
     *   PMEAN(5),PMIN2(5),PVAR(5),PSDEV(5),
     *   IHALT(4),IHPMIN(5,52),
```

```
     *    UTIL(1),RUTIL(1),CERTEQ(1),
     *    PMIN(5),PRMIN(5),PRMAX(5),PMTX(4,5),
     *    PRICEZ(5,5),CDF(5,52),PCDF(5,52),ZPCT(5)

***    {END MODULE 3}


***** DEFINE COMMON BLOCKS.  {MODULE 4}

      COMMON
     *    /COM00/IDATA,JDIAG,JLOUT,NTRIAL,NIA,NIX,NIT,NID,NIP,NII,NIC,NIU
     *    /COM01/ITASK,TDATA,TCOEF
     *    /COM02/RNSEED,RANDOM,RANMTX
     *    /COM03/UDATA,UCOEF,CECOEF
     *    /COM04/PLB,PUB
     *    /COM05/PMEAN,PMIN2,PVAR,PSDEV
     *    /COM06/IHALT,IHPMIN
     *    /COM07/IITR
     *    /COM08/UTIL,RUTIL,CERTEQ
     *    /COM09/PMIN,PRMIN,PRMAX,PMTX
     *    /COM10/PRICEZ,CDF,PCDF,ZPCT

***    {END MODULE 4}


***** EQUIVANCE ARRAYS.  {MODULE 5}

*      RANMTX AND VARMTX CAN BE EQUIVALENCED BECAUSE THE ELEMENTS OF
*      VARMTX ARE CALCULATED FROM THE ELEMENTS OF RANMTX ON EACH MONTE
*      CARLO TRIAL ON A ONE-FOR-ONE ELEMENT BASIS.

      EQUIVALENCE (RANMTX,VARMTX)

***    {END MODULE 5}

************************** INITIAL.FOR ***********************************
```

```
C**********************************************************************
C                            EQNS.FOR                                  *
C                                                                      *
C  PROGRAM: SIMRAND I                                                  *
C  DATE:    12/05/84                                                   *
C                                                                      *
C  THIS ROUTINE IS TO BE 'INCLUDED' IN THE SUBROUTINE PRICE.FOR IN     *
C  THE SIMRAND I PROGRAM IMPLEMENTATION FOR THE LARGE SCALE SYSTEMS    *
C  ARTICLE.                                                            *
C                                                                      *
C----------------------------------------------------------------------*
C                        CONFIGURATION CHANGES                         *
C                                                                      *
C    DATE                       CHANGE                                *
C                                                                      *
C  12/05/84   * ORIGINAL.                                              *
C                                                                      *
C**********************************************************************


*****       CALCULATE STEP PRICES PMTX(IA,IP).  {MODULE 4}

          CI = 1000.0
          CD = 2330.0
          CU = 2.54E-5

          NIXX = NIX - 1
          DO 2000 IX=1,NIXX
             X(IX) = VARMTX(IA,IX)
2000      CONTINUE

          PMTX(IA,1) = (X(1)*X(2)*CU*CD) / (CI*X(3))

          IF ((IA .EQ. 1) .OR. (IA .EQ. 3))
     *       PMTX(IA,2) = (X(4)*X(2)*CU*CD) / (CI*X(3))

          IF ((IA .EQ. 2) .OR. (IA .EQ. 4))
     *       PMTX(IA,2) = X(4) / (CI*X(3))

          IF ((IA .EQ. 1) .OR. (IA .EQ. 3))
     *       PMTX(IA,3) = X(5) / (CI*X(3))

          IF ((IA .EQ. 2) .OR. (IA .EQ. 4))
     *       PMTX(IA,3) = 0.0

          PMTX(IA,4) = X(6) / (CI*X(3))

***       {END MODULE 4}


*****       CALCULATE TOTAL PRICE PMTX(IA,NIP).  {MODULE 5}

          PMTX(IA,NIP) = 0.0
```

```
            NIPP = NIP - 1
            DO 2010 IP=1,NIPP
               PMTX(IA,NIP) = PMTX(IA,NIP) + PMTX(IA,IP)
2010        CONTINUE

***         {END MODULE 5}
```

APPENDIX B

INPUT DATA FILE

File: DATAIN.DAT

APPENDIX B
Input Data File

```
IDATA JDIAG JLOUT NTRIAL NIA NIX NIT NID  NIP  NII
    1     1     0   1000   4   7  10  16    5    1
RANDOM NUMBER SEED (RNSEED)
1.0D7
   IA              ITASK(IA,IX)
    1          1    2    4    6    8    9    1
    2          1    3    5    7    0    9    1
    3          1    2    4    6    8   10    1
    4          1    3    5    7    0   10    1
   IT              TDATA(IT,ID)
  SILICON (All)
    1       40.000         0.800         8.000         0.000
            18.000         1.000         0.000         0.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  T (Ingot)
    2       23.000         1.000        14.000         0.000
            23.000         1.000         0.000         0.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  T (Ribbon)
    3       10.000         1.000         2.000         0.000
            10.000         1.000         0.000         0.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  ne (Ingot)
    4        0.130         1.000         0.130         0.000
             0.160         1.000         0.000         0.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  ne (Ribbon)
    5        0.100         1.000         0.100         0.000
             0.130         0.750         0.150         1.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  CRYSTAL (Ingot)
    6       30.000         0.900        13.000         0.000
            17.000         1.000         0.000         0.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  CRYSTAL (Ribbon)
    7       50.000         0.800         4.000         0.000
            10.000         0.500        25.000         1.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  SAW (Ingot)
    8       40.000         0.800         8.000         0.000
            24.000         1.000         0.000         0.000
             0.000         0.000         0.000         0.000
             0.000         0.000         0.000         0.000
  CELL (4A)
    9       25.000         1.000        12.000         0.000
            14.000         0.100        15.000         0.250
            16.000         0.500        18.000         0.750
```

File: DATAIN.DAT

```
          22.000          0.900          25.000          1.000
  CELL (4B)
   10     25.000          1.000          12.000          0.000
          25.000          1.000           0.000          0.000
           0.000          0.000           0.000          0.000
           0.000          0.000           0.000          0.000
    II              UDATA(II,ID)
    1      0.25           1.00            1.25           0.00
           0.00           0.00            0.00           0.00
           0.00           0.00            0.00           0.00
           0.00           0.00            0.00           0.00
LOWER BOUND ON PRICE RANGES PLB(IP)
           0.00           0.00            0.00           0.00           0.00
UPPER BOUND ON PRICE RANGES PUB(IP)
           0.50           0.50            0.50           0.50           1.00
```

APPENDIX C

OUTPUT DATA FILE

File: DATAOUT.DAT

APPENDIX C
Output Data File

IDATA = 1 NTRIAL = 1000 RANDOM SEED = 10000000.

PMEAN	6.3475E-02	1.2111E-01	1.5625E-02	1.2768E-01	3.2789E-01
PSDEV	4.8853E-02	8.6448E-02	4.2896E-02	2.5607E-02	1.1705E-01
PRMIN	8.0358E-03	2.7265E-02	.0000E+00	7.6989E-02	1.3831E-01
.10	3.0000E-02	5.0000E-02	.0000E+00	1.0000E-01	2.2000E-01
.25	4.0000E-02	7.0000E-02	.0000E+00	1.1000E-01	2.4000E-01
.50	5.0000E-02	1.0000E-01	.0000E+00	1.3000E-01	3.2000E-01
.75	8.0000E-02	1.5000E-01	.0000E+00	1.5000E-01	4.0000E-01
.90	1.4000E-01	2.0000E-01	9.0000E-02	1.7000E-01	5.0000E-01
PRMAX	3.2612E-01	4.9416E-01	2.9056E-01	2.3452E-01	8.3985E-01
RUTIL	.9113				
CERTEQ	.3387				

IHALT 95 512 43 350
RANDOM NUMBER SEED (RNSEED) FOR NEXT RUN = 15969712.

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. 85-96	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle The SIMRAND I Computer Program		5. Report Date February 15, 1986 6. Performing Organization Code
7. Author(s) R. F. Miles, Jr.		8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109		10. Work Unit No. 11. Contract or Grant No. NAS7-918 13. Type of Report and Period Covered JPL Publication
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		14. Sponsoring Agency Code

15. Supplementary Notes Sponsored by the U.S. Department of Energy through Interagency Agreement DE-AI01-85CE89008 with NASA; also identified as DOE/JPL 1012-115 and as JPL Project No. 5101-274 (RTOP or Customer Code 776-52-61).

16. Abstract

This report documents the SIMRAND I Computer Program (Version 5.0x0 3) written in Microsoft FORTRAN for the IBM PC microcomputer and its compatibles. The SIMRAND I Computer Program comprises eleven modules-a main routine and ten subroutines. Two additional files are used at compile time; one inserts the system or task equations into the source code, while the other inserts the dimension statements and common blocks. The SIMRAND I Computer Program can be run on most microcomputers or mainframe computers with only minor modifications to the computer code.

17. Key Words (Selected by Author(s))	18. Distribution Statement
Computer Programming and Software Systems Analysis	Unclassified-Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	90	